Zur Lösung von zahlentheoretischen Problemen mit klassischen und Quantencomputern

Vom Fachbereich Informatik
der Technischen Universität Darmstadt
genehmigte

Dissertation

zur Erreichung des akademischen Grades
Doctor rerum naturalium (Dr. rer. nat.)

von

Dipl.-Inform. Arthur Schmidt

aus Presnogorkowka

Referenten:	Prof. Dr. Johannes Buchmann Prof. Dr. Michael John Jacobson, Jr.
Tag der Einreichung:	12.März 2007
Tag der mündlichen Prüfung:	08.Mai 2007

Darmstadt, 2007
Hochschulkennziffer: D 17

Erklärung[1]

Hiermit erkläre ich, dass ich die vorliegende Arbeit – abgesehen von den in ihr ausdrücklich genannten Hilfen – selbständing verfaßt habe.

Wissenschaftlicher Werdegang des Verfassers in Kurzfassung[2]

Oktober 1996 – Januar 2002	Studium der Informatik an der Technischen Universität Darmstadt
Januar 2002	Diplomabschluß (Diplom-Informatiker)
Mai 2002 – Dezember 2005	Wissenschaftlicher Mitarbeiter am Fachgebiet Theoretische Informatik, Fachbereich Informatik, Technische Universität Darmstadt

[1] gemäß §9 Abs. 1 der Promotionsordnung der TU Darmstadt
[2] gemäß §20 Abs. 3 der Promotionsordnung der TU Darmstadt

An dieser Stelle möchte ich mich bei den Menschen bedanken, die mich während meiner Promotion unterstützt haben.

In erster Linie möchte ich Prof. Johannes Buchmann herzlich dafür danken, daß er bei mir das Interesse an der algorithmischen Zahlentheorie geweckt hat und mir die Gelegenheit gab, an seinem Lehrstuhl zu promovieren. Seine Vorschläge und Diskussionen mit ihm waren stets sehr wertvoll.

Ich bedanke mich bei Prof. Michael Jacobson für die Übernahme der Zweitkorrektur und für die wertvollen Kommentare, die zur Verbesserung dieser Dissertation geführt haben.

Ein besonderer Dank geht an Dr. Ulrich Vollmer für die vielen konstruktiven Diskussionen, Ideen und Vorschläge.

Marita Skrobic danke ich für die großartige Hilfe bei administrativen Aufgaben.

Ich danke Dr. Ulrike Mayer für die Korrektur von Teilen dieser Dissertation und allen Mitarbeitern des Fachgebiets für die nette Atmosphäre und eine sehr angenehme Zeit.

Ich danke der Deutschen Forschungsgemeinschaft für die finanzielle Unterstützung während meiner Promotion.

Und natürlich möchte ich mich bei meiner Familie und insbesondrere meiner Frau Kathrin für viel Unterstützung und die große Geduld herzlichst bedanken.

Zusammenfassung

In dieser Arbeit werden Algorithmen zur Lösung zahlentheoretischer Probleme für klassische Computer und Quantencomputer entwickelt und analysiert sowie ihre Auswirkungen auf Public-Key-Kryptosysteme untersucht.

Im Jahr 1994 veröffentliche Shor einen Quantenalgorithmus [Sho94], der das Periodengitter einer gegebenen Funktion in Quanten-Polynomzeit bestimmt. Mit diesem Algorithmus läßt sich das Faktorisierungs- und das Diskreter-Logarithmus-Problems in endlichen abelschen Gruppen in Quanten-Polynomzeit lösen. Dadurch werden die meisten heutzutage verwendeten Public-Key-Kryptoverfahren unsicher, sollten hinreichend große Quantencomputer eines Tages gebaut werden können.

Der erste Quantencomputer wurde 1998 gebaut [CVZ$^+$98]. Er bestand aus zwei Qubits. Der größte Quantencomputer, der bis heute gebaut wurde, besteht aus nur sieben Qubits [VSB$^+$01]. Es hat sich herausgestellt, daß das Bauen von großen Quantencomputern ein extrem schwieriges Problem ist. Aus diesem Grund kann davon ausgegangen werden, daß die Größe der Quantencomputer, d.h. die Anzahl der Qubits, aus denen ein Quantencomputer besteht, nur langsam wachsen wird. Deshalb ist es von Interesse, Shors Quantenalgorithmus an verschiedene abelsche Gruppen sowie gruppenähnliche Strukturen zu adaptieren und ihre genaue Komplexität zu untersuchen, um daraus die Bedrohungen für die betroffenen Kryptosysteme genau zu bestimmen.

In dieser Arbeit werden Quantenalgorithmen zur Lösung des Ordnungsproblems und des DL-Problems in der Klassengruppe der Ideale eines imaginär-quadratischen Zahlkörpers angegeben und ihre Komplexität, d.h. die Anzahl der Qubits und der elementaren Quantenoperationen, bestimmt. Darüberhinaus werden Algorithmen zur Bestimmung des Regulators und zur Lösung des Hauptidealproblems und des erweiterten Diskreter-Logarithmus-Problems in reell-quadratischen Zahlkörpern präsentiert. Shors Algorithmus wird dabei auf Funktionen angewendet, die nicht injektiv innerhalb einer Periode sind. Auch in diesem Fall wird die Komplexität der Algorithmen genau bestimmt.

Im Falle von Zahlkörpern eines beliebigen Grades wird eine weitere Modifikation von Shors Algorithmus entwickelt: Es wird das Periodengitter einer Funktion bestimmt, deren Periodengitter irrational ist. Dieser Algorithmus wird benutzt, um die Einheitengruppe eines beliebigen endlich-dimensionalen Zahlkörpers zu berechnen.

Als nächstes werden die Auswirkungen von Quantenalgorithmen auf die Sicherheit von Public-Key-Kryptosystemen untersucht. Es wird ein neuer Sicherheitsbegriff eingeführt. Die Sicherheit zweier Kryptosysteme wird als gleich definiert, wenn die Anzahl der Qubits zum Brechen dieser Kryptosysteme gleich ist. Basierend auf der neuen Sicherheitsdefinition wird die Geschwindigkeit und die Schlüsselgröße von einigen Kryptosystemen verglichen.

Im letzten Kapitel wird ein deterministischer klassischer Algorithmus zur Berechnung der Struktur einer endlichen abelschen Gruppe aus einem Erzeugendensystem präsentiert, dessen Laufzeit nur linear von der Anzahl der gegebenen Erzeuger abhängt und dessen Speicherplatzkomplexität unabhängig von der Anzahl der gegebenen Erzeuger ist. Das stellt eine exponentielle Verbesserung zu früheren Algorithmen dar, bei denen sowohl die Laufzeit als auch die Speicherplatzkomplexität exponentiell von der Anzahl der Erzeuger im gegebenen Erzeugendensystem abhängt [BJT97].

Abstract

In this thesis, we will present and analyze algorithms for classical and quantum computers which solve some number theoretical problems. Moreover we will investigate their impact on current public key cryptosystems.

In [Sho94] Shor presented quantum algorithms which determine the period lattice of some functions in quantum polynomial time. These algorithms can be applied to solve the factoring problem and the discrete logarithm problem in finite abelian groups in quantum polynomial time. Therefore, if large quantum computers are built in the future, then almost all public key cryptosystems which are used today will be broken.

The first quantum computer was built 1998 [CVZ+98]. It had two qubits. Until now, the largest quantum computer has only seven qubits [VSB+01]. Building large quantum computers seems to be an extremely difficult task. Therefore we can assume that the size of quantum computers, i.e. the number of qubits, will grow very slowly. Thus it is of interest to adapt Shor's algorithms to different abelian groups and group-like structures and to analyze their complexity closely.

In this thesis, we will present algorithms which solve the order and the discrete logarithm problem in the class group of an imaginary quadratic number field. We will analyze their complexity and determine the number of qubits and elementary quantum gates which are necessary to implement these algorithms. Moreover we will present algorithms for computing the regulator and solving the principal ideal and the extended discrete logarithm problem in real quadratic number fields. Thereby, we will apply Shor's algorithm to functions which are not injective inside a period. As in the imaginary quadratic case, we will give an accurate estimation of the complexity of the presented algorithms.

In the case of number fields of an arbitrary degree we will develop a further modification of Shor's algorithm. The new algorithm computes the period lattice on a function which has an irrational period lattice. We will apply this algorithm to compute the unit group of finite extensions of $\mathbb{Q}$.

Next, we will investigate the impact of quantum algorithms on current public key cryptosystems. We will define a new security parameter: the number of qubits which are necessary to implement an attack. Based on this new definition we will compare the speed and key sizes of some cryptosystems.

In the last chapter we will present a classical algorithm that computes the structure of a finite abelian group from a generating system. Its runtime depends only linearly on the number of given generators. Its space requirements don't depend on the number of generators at all. This is an exponential improvement. Both runtime and space complexity of prior algorithms depend exponential on the number of generators in the given generating system [BJT97].

Inhaltsverzeichnis

Liste der Algorithmen

Tabellenverzeichnis

Kapitel 1

Einleitung

Public-Key-Kryptosysteme spielen eine immer größere Rolle in unserer Gesellschaft. Im Zeitalter der elektronischen Kommunikation sind sie ein Mittel, um die Vertraulichkeit, Integrität und Verbindlichkeit zu gewährleisten. Seit ihrer Entdeckung Ende der 70er Jahre wurden mehrere Kryptosysteme vorgeschlagen und hinsichtlich ihrer Sicherheit intensiv untersucht.

Die Sicherheit der Public-Key-Kryptosysteme basiert auf der Schwierigkeit der Lösung unterschiedlicher mathematischer Probleme. Die meisten Kryptosysteme, die heutzutage praktisch eingesetzt werden, beziehen ihre Sicherheit aus dem Faktorisierungsproblem oder dem Disktreter-Logarithmus-Problem in endlichen abelschen Gruppen. Diese Probleme gelten als schwierig, da bis jetzt noch keine Polynomzeit-Algorithmen für klassische Computer zu ihrer Lösung gefunden wurden.

Im Jahr 1994 präsentierte P. Shor Algorithmen für Quantencomputer, die sowohl das Faktorisierungsproblem als auch das DL-Problem in endlichen abelschen Gruppen in Polynomzeit lösen. Zur Zeit stellen diese Algorithmen keine ernsthafte Bedrohung für die Public-Key-Kryptographie dar, da das Bauen hinreichend großer Quantencomputer eine extrem schwierige Aufgabe zu sein scheint. Jedoch sind Quantencomputer schon längst keine rein theoretische Idee mehr: Der größte Quantencomputer, auf dem Shors Algorithmus implementiert wurde, besteht aus sieben Qubits. Der größte Quantenspeicher kann ein ganzes Quanten-Byte speichern. Viele Forscher erwarten, daß in Zukunft auch größere Quantencomputer gebaut werden können. Wenn dieser Fall eintritt, werden die meisten heutzutage eingesetzten Kryptosysteme unsicher.

Um von dieser Entwicklung nicht überrannt zu werden, müssen Kryptographen bereits heute die Auswirkungen von Quantencomputern sehr genausten untersuchen und geeignete Gegenmaßnahmen entwickeln. Zwei Alternativen bieten sich als Gegenmaßnahmen an: Die erste Alternative ist, die Sicherheitsparameter bestehender Kryptoverfahren zu erhöhen, die zweite Alternative besteht darin, alternative Kryptoverfahren einzusetzen, deren Sicherheit auf Problemen basiert, für die bis heute keine Quanten-Polynomzeitalgorithmen bekannt sind. Solche Probleme kommen z.B. aus der Gittertheorie, der Kodierungstheorie oder der Theorie der quadratischen multivariaten Gleichungssystemen. Die erste Alternative kann in der Übergangszeit verwendet werden, solange die Quantencomputer nicht zu groß sind (bis ca. 2^{19} Qubits). Sollten noch größere Quantencomputer gebaut werden, ist der Einsatz alternativer Kryptoverfahren empfehlenswert.

In der vorliegenden Arbeit wird der erste Ansatz verfolgt. Aufbauend auf den Arbeiten von Beauregard [Bea02], Proos und Zalka [PZ03], die Speicherplatz-optimierte Quantenalgorithmen zur Lösung des Faktorisierungs- und des DL-Problems in der Punktegruppe elliptischer Kurven entwickelten, werden hier Algorithmen für quadratische Zahlkörper vorgestellt. Es werden Auswirkungen dieser Algorithmen auf die betreffende Kryptosysteme untersucht und ihre Effizienz verglichen.

In dieser Arbeit werden zwei weitere Themen behandelt, die mit der Kryptographie nur am Rande etwas zu tun haben. Es wird ein Quantenalgorithmus zur Lösung des Hidden-Subgroup-Problems in $\mathbb{R}^n$ präsentiert. Darauf aufbauend wird ein Algorithmus zur Bestimmung der Einheitengruppe eines beliebigen Zahlkörper vorgestellt. Die Laufzeit dieses Algorithmus hängt dabei exponentiell vom Körpergrad und polynomiell von der Diskriminante des Körpers ab. Ein weiterer Algorithmus wird im achten Kapitel präsentiert. Er läuft komplett auf klassischen Computern und bestimmt die Struktur einer endlichen abelschen Gruppe.

1.1 Vorarbeiten

In [Sho94] präsentierte Shor einen Quantenalgorithmus, welcher in Polynomzeit das Faktorisierungs- und das DL-Problem in endlichen abelschen Gruppen löst. Diese Algorithmen wurden mehrfach hinsichtlich ihres Qubit-Bedarfs optimiert. Der bis jetzt beste Algorithmus wurde von Beauregard in [Bea02] vorgeschlagen. Er benötigt $2n + 3$ Qubits, um eine n-Bit Zahl zu faktorisieren. Zur Reduktion der Qubits verwendet Beauregards Algorithmus die semi-klassische Quantenfouriertransformation [GN96], sowie die Quantenaddition [Dra00].

Das DL-Problem in der Punktegruppe einer elliptischen Kurve über $\mathbb{F}_p$ untersuchten Proos und Zalka in [PZ03]. Sie zeigten, daß für das Lösen des DL-Problems ca. $7 \log_2 p + 4 \log_2 \log_2 p$ Qubits notwendig sind. Unter plausiblen, jedoch nicht bewiesenen Annahmen, konnten sie dieses Ergebnis auf $5 \log_2 p + 8(\log_2 p)^{1/2} + 4 \log_2 \log_2 p$ Qubits reduzieren.

In [Hal02] untersuchte Hallgren Algorithmen zur Berechnung des Regulators eines reell-quadratischen Zahlkörpers und zur Lösung des Hauptidealproblems in reell-quadratischen Zahlkörpern. Zu diesem Zweck schlug er zwei periodische Funktionen vor, und bestimmte ihre Perioden. Daraus leitete er die Lösungen für die oben genannten Probleme ab. Die Laufzeit und der Speicherbedarf der Algorithmen wurde als polynomiell angegeben. Die von Hallgren vorgeschlagenen Funktionen können jedoch streng genommen nicht in Polynomzeit berechnet werden, da sie intern natürliche Logarithmen von algebraischen Zahlen ausrechnen müssen. Deshalb sind die Aussagen über die polynomielle Laufzeit nicht ganz korrekt.

Auf der klassischen Seite wurden Berechnungsprobleme in Zahlkörpern ausführlich von Thiel in [Thi95] behandelt.

Buchmann und Pohst schlugen in [BP89] ein Algorithmus zur Berechnung einer Approximation einer Basis eines reellen Gitters vor. Als Eingabe bekommt dieser Algorithmus Approximationen von Vektoren, welche das Gitter erzeugen. Eine zusätzliche Analyse des Algorithmus wurde in [BK93] durchgeführt.

Ein klassischer Algorithmus zur Bestimmung der Struktur einer endlichen abelschen Gruppe wurde von Buchmann, Jacobson und Teske in [BJT97] vorgeschlagen. Der Algorithmus be-

kommt als Eingabe ein Erzeugendensystem $\mathcal{M}$ einer Gruppe $\mathcal{G}$ und berechnet die Struktur von $\mathcal{G}$. Die Laufzeit und der Speicherplatzbedarf des Algorithmus ist $O(2^{|\mathcal{M}|}\sqrt{|\mathcal{G}|})$.

1.2 Eigene Resultate

In der vorliegenden Arbeit werden Algorithmen zur Lösung des Ordnungs- und des Diskreter-Logarithmus-Problems in der Klassengruppe der Ideale imaginär-quadratischer Zahlkörper präsentiert und analysiert. Dazu werden Quantenalgorithmen zur Multiplikation und Reduktion von Idealen entwickelt und eine präzise Abschätzung für die Anzahl benötigter Qubits und elementarer Quantengatter berechnet. Zur Lösung des Ordnungs- und DL-Problems wird das Quantenframework zur Bestimmung der Perioden von periodischen Funktionen verwendet (siehe Seite 58).

Im Fall reell-quadratischer Zahlkörper werden Algorithmen zur Bestimmung des Regulators und zur Lösung des Hauptideal-, des Ordnungs- und des erweiterten DL-Problems vorgeschlagen und analysiert. Zu diesem Zweck werden Algorithmen zur Reduktion von Idealen, zur Bestimmung der Nachbarn auf dem Reduktionszyklus und zur Berechnung der Abstände zwischen zwei Idealen entwickelt. Die neuen Algorithmen enthalten nicht mehr den Fehler aus Hallgrens Arbeit. Sie führen jedoch dazu, das das Quantenframework auf Funktionen angewendet wird, welche innerhalb einer Periode nicht injektiv sind. Es wird gezeigt, daß auch in diesem Fall das Quantenframework mit großer Wahrscheinlichkeit die Periode richtig berechnet.

Zur Bestimmung der Einheitengruppe einer endlichen Körpererweiterung von $\mathbb{Q}$ entwickeln wir einen Algorithmus, welcher eine Approximation einer Basis für das Periodengitter $L \subset \mathbb{R}^r$ einer schwach-periodischen Funktion (siehe Definition 6.1.1) berechnet. Dazu wird das Quantenframework so modifiziert, daß es Approximationen von Vektoren aus L^* ausgibt. Es wird die Wahrscheinlichkeit bestimmt, daß $2r$ solcher Vektoren das ganze Gitter L^* erzeugen. Schließlich wird der Algorithmus von Buchmann und Pohst [BP89] angepaßt, so daß er aus den vorliegenden $2r$ Approximationen von Vektoren aus L^* eine Approximation einer Basis von L berechnet.

Motiviert durch die Annahme, daß das Bauen großer Quantencomputer extrem schwierig ist, wird ein neuer Begriff zur Sicherheitsäquivalenz von Kryptosystemen definiert: die Anzahl der Qubits, welche zum Brechen des Kryptosystems nötig sind. Basierend auf den Arbeiten von Beauregard [Bea02], Proos und Zalka [PZ03] und den Ergebnissen aus dem vierten und fünften Kapitel vorliegender Arbeit werden hier einige klassische Kryptoverfahren hinsichtlich ihrer Laufzeit und ihrer Schlüsselgröße miteinander verglichen.

Schließlich wird ein klassischer Algorithmus zur Bestimmung der Struktur einer endlichen abelschen Gruppe präsentiert. Die Laufzeit des neuen Algorithmus hängt nur linear von der Länge des Erzeugendensystems ab, der Speicherbedarf ist davon unabhängig. Das ist eine exponentielle Verbesserung der Ergebnisse aus [BJT97].

Kapitel 2

Mathematische Grundlagen

In diesem Kapitel werden mathematische Grundlagen und grundlegende Algorithmen präsentiert, welche in den späteren Kapiteln verwendet werden. Dabei werden bei bereits bekannten Sätzen die Beweise nicht angegeben. Die neuen Sätzen, welche beim Schreiben dieser Arbeit entstanden sind, werden dagegen bewiesen.

Das Kapitel ist wie folgt aufgebaut: Zuerst werden die grundlegenden Definitionen und Sätze der Gruppentheorie präsentiert. Im darauf folgenden Abschnitt wird die Wahrscheinlichkeit dafür berechnet, daß zwei ganze Zahlen, welche zufällig aus einer gewissen Menge gewählt werden, teilerfremd sind. Im dritten Abschnitt wird die Definition eines Kettenburchs und ein Resultat über die Kettenbruchentwicklung angegeben. Im nächsten Abschnitt werden die nötigen Grundlagen der Gittertheorie präsentiert. Außerdem werden dort wichtige Resultate über die Wahrscheinlichkeit für Gittererzeugung und über die Approximationsschranken eines Algorithmus zum Finden einer Basis des dualen Gitters bewiesen. Im letzen Kapitel werden die Grundlagen der Zahlentheorie behandlt.

2.1 Endliche abelsche Gruppen

In diesem Abschnitt präsentieren wir die nötigen Grundbegriffe und Sätze aus der Gruppentheorie. Für eine ausführliche Einführung in das Thema verweisen wir auf [Art93], [Lan02] und andere Standardbücher über Algebra.

Satz 2.1.1 (Gruppe) *Sei $\mathcal{G}$ eine Menge und $\cdot : \mathcal{G} \times \mathcal{G} \to \mathcal{G}$ eine assoziative Abbildung, so daß*

- *ein Element $e \in \mathcal{G}$ mit $g \cdot e = g = e \cdot g$ für alle $g \in \mathcal{G}$ existiert und*
- *für jedes $g \in \mathcal{G}$ ein Element h mit $g \cdot h = e = h \cdot g$ existiert.*

Das Tripel $(\mathcal{G}, \cdot, e)$ ist eine Gruppe*. Das Gruppenelement e heißt das* neutrale Element*. Das Element h ist das Inverse von g. Für das Inverse von g schreiben wir auch g^{-1}.*

Für das Tripel $(\mathcal{G}, \cdot, e)$ schreiben wir abkürzend $\mathcal{G}$. Mit $|\mathcal{G}|$ oder $\operatorname{card} \mathcal{G}$ bezeichnen wir die Anzahl der Gruppenelemente. Wir lassen oft das Zeichen $\cdot$ weg und schreiben gh anstatt $g \cdot h$.

Falls die Abbildung kommutativ ist, heißt die Gruppe abelsch *oder* kommutativ. *Die Gruppe heißt endlich, falls $\mathcal{G}$ eine endliche Menge ist.*

Satz 2.1.2 *Seien $l \in \mathbb{N}$, $\mathcal{G}$ eine endliche abelsche Gruppe und $g_1, \ldots, g_l \in \mathcal{G}$. Wir setzen*

$$\langle g_1, \ldots, g_l \rangle := \{ \prod_{i=1}^{l} g_i^{e_i} \mid e_i \in \mathbb{Z}, 1 \leq i \leq l \}.$$

Satz 2.1.3 (Untergruppe) *Eine Teilmenge $\mathcal{H}$ einer Gruppe $\mathcal{G}$ heißt* Untergruppe, *falls die folgenden Bedingungen erfüllt sind:*

- *Ist $g \in \mathcal{H}$ und $h \in \mathcal{H}$, dann ist auch $gh \in \mathcal{H}$,*
- *$e \in \mathcal{H}$ und*
- *ist $g \in \mathcal{H}$, dann ist auch das Inverse $g^{-1} \in \mathcal{H}$.*

Satz 2.1.4 (Nebenklasse) *Seien $\mathcal{G}$ eine abelsche Gruppe, $\mathcal{H}$ eine Untergruppe von $\mathcal{G}$ und $g \in \mathcal{G}$. Dann ist*

$$g\mathcal{H} = \{ gh \mid h \in \mathcal{H} \}$$

eine Nebenklasse *von $\mathcal{H}$.*

Das Produkt zweier Teilmengen $\mathcal{X}$ und $\mathcal{Y}$ einer abelschen Gruppe ist die Menge $\mathcal{XY} = \{ \mathsf{gh} \mid \mathsf{g} \in \mathcal{X},\ \mathsf{h} \in \mathcal{Y} \}$.

Satz 2.1.5 *Sei $\mathcal{H}$ eine Untergruppe einer abelschen Gruppe $\mathcal{G}$. Das Produkt von zwei Nebenklassen $g_1\mathcal{H}$ und $g_2\mathcal{H}$ ist die Nebenklasse $(g_1g_2)\mathcal{H}$. Die Menge der Nebenklassen mit oben definierter Multiplikation bildet eine abelsche Gruppe.*

Satz 2.1.6 (Faktorgruppe) *Die Gruppe aus dem letzten Satz nennen wir die Faktorgruppe und bezeichnen diese mit $\mathcal{G}/\mathcal{H}$.*

Satz 2.1.7 (direktes Produkt) *Seien $\mathcal{G}_1, \cdots, \mathcal{G}_k$ Gruppen. Auf der Menge*

$$\mathcal{G}_1 \times \cdots \times \mathcal{G}_k = \{ (g_1, \ldots, g_k) \mid g_i \in \mathcal{G}_i,\ 1 \leq i \leq k \}$$

läßt sich durch die Operation

$$(g_1, \ldots, g_k) \cdot (h_1, \ldots, h_k) = (g_1 \cdot_{\mathcal{G}_1} h_1, \ldots, g_k \cdot_{\mathcal{G}_k} h_k)$$

eine Gruppenstruktur definieren, wobei $\cdot_{\mathcal{G}_i}$ die Gruppenoperation in der Gruppe $\mathcal{G}_i$, $1 \leq i \leq k$, ist. Diese neue Gruppe nennen wir das direkte Produkt *von $\mathcal{G}_1, \ldots, \mathcal{G}_k$ und Bezeichnen es mit $\mathcal{G}_1 \times \cdots \times \mathcal{G}_k$.*

Satz 2.1.8 *Zwei Gruppen $\mathcal{G}_1$ und $\mathcal{G}_2$ heißen* isomorph, *falls ein Isomorphismus $\phi : \mathcal{G}_1 \to \mathcal{G}_2$ existiert, d.h. ϕ ist eine bijektive Abbildung mit der Eigenschaft, daß $\phi(gh) = \phi(g)\phi(h)$ für alle $g, h \in \mathcal{G}_1$. Wenn $\mathcal{G}_1$ und $\mathcal{G}_2$ isomorph sind, dann schreiben wir $\mathcal{G}_1 \cong \mathcal{G}_2$.*

Satz 2.1.9 (Erzeugendensystem) *Sei $\mathcal{G}$ eine endliche abelsche Gruppe. Die Folge $(g_1, \ldots, g_l) \in \mathcal{G}^l$ heißt ein Erzeugendensystem von $\mathcal{G}$, wenn gilt:*

$$\mathcal{G} = \langle g_1, \ldots, g_l \rangle.$$

Es gilt der folgende Struktursatz.

Satz 2.1.10 (Struktursatz für endliche abelsche Gruppen) *Sei $\mathcal{G}$ eine endliche abelsche Gruppe, dann gibt es natürliche Zahlen k und $n_1, \ldots, n_k$, mit $n_1 > 1$ und n_{i+1} teilt n_i für $1 \leq i < k$, und zyklische Untergruppen $\mathcal{G}_1, \ldots, \mathcal{G}_k$, mit $|\mathcal{G}_i| = n_i$, $1 \leq i \leq k$, so daß gilt*

$$\mathcal{G} \cong \mathcal{G}_1 \times \cdots \times \mathcal{G}_k.$$

Dabei sind $n_1, \ldots, n_k$ eindeutig bestimmt. Sie heißen Invarianten der Gruppe G.

Ein Beweis dieses Satzes kann in jedem Standardbuch über Algebra gefunden werden.

Wir definieren nun das Strukturproblem.

Problem 2.1.11 (Das Strukturproblem) *Gegeben sei ein Erzeugendensystem einer endlichen abelschen Gruppe $\mathcal{G}$. Bestimme die Invarianten $n_1, \ldots, n_k$ von $\mathcal{G}$ und Gruppenelemente $g_1, \ldots, g_k \in \mathcal{G}$, so daß $\mathcal{G} \cong \langle g_1 \rangle \times \cdots \times \langle g_k \rangle$ und $|\langle g_i \rangle| = n_i$ für $1 \leq i \leq k$.*

Einen Algorithmus zur Lösung dieses Problem beschreiben wir im Kapitel 8.

Der folgende Satz bestimmt eine Untergrenze für die Wahrscheinlichkeit, mit der ein zufälliges Tupel ein Erzeugendensystem einer endlichen abelschen Gruppe bildet.

Satz 2.1.12 *Sei $\mathcal{G}$ eine endliche abelsche Gruppe, dann gilt*

$$\frac{\mathrm{card}\{\, (g_1, \ldots, g_{r+1}) \in \mathcal{G}^{r+1} \mid \langle g_1, \ldots, g_{r+1} \rangle = \mathcal{G} \,\}}{|\mathcal{G}|^{r+1}} > \frac{1}{4}.$$

Beweis. Siehe Theorem 2.2 in [PB99].

2.2 Wahrscheinlichkeit für Teilerfremdheit

In diesem Abschnitt präsentieren wir eine untere Grenze für die Wahrscheinlichkeit, daß zwei zufällig gewählte Zahlen teilerfremd sind.

Ein wohlbekanntes Resultat besagt

Satz 2.2.1 *Sei $n \in \mathbb{N}$, dann gilt*

$$\frac{\mathrm{card}\{\, (a,b) \in \mathbb{N}^2 \mid 1 \leq a, b < n \text{ und } \gcd(a,b) = 1 \,\}}{\mathrm{card}\{\, (a,b) \in \mathbb{N}^2 \mid 1 \leq a, b < n \,\}} \geq \frac{6}{\pi^2}$$

Beweis. Siehe z.B. [Knu81].

Wir verallgemeinern diesen Satz auf beliebige Zahlenabschnitte. Dazu beweisen wir zuerst das folgende Lemma

Lemma 2.2.2 *Seien $c \in \mathbb{Z}$ und $d \in \mathbb{N}$, dann gilt*

$$\frac{\operatorname{card}\{\,(a,b) \in \mathbb{Z}^2 \mid 0 \le a,b \le d \text{ und } \gcd(a,b) = 1\,\} - 1}{\operatorname{card}\{\,(a,b) \in \mathbb{Z}^2 \mid c \le a,b \le c+d \text{ und } \gcd(a,b) = 1\,\}} \le 1$$

Beweis. Seien $c \in \mathbb{Z}$, $d \in \mathbb{N}$. Wir definieren die folgenden Mengen

- $\mathcal{A} = \{\,(a,b) \in \mathbb{Z}^2 \mid 0 \le a,b \le d\,\}, \quad \mathcal{B} = \{\,(a,b) \in \mathbb{Z}^2 \mid c \le a,b \le c+d\,\},$
- $\mathcal{A}' = \{\,(a,b) \in \mathcal{A} \mid \gcd(a,b) = 1\,\}, \quad \mathcal{B}' = \{\,(a,b) \in \mathcal{B} \mid \gcd(a,b) = 1\,\}$
- $\mathcal{A}_p = \{\,(a,b) \in \mathcal{A} \mid a \ne b \;\&\; p \mid \gcd(a,b)\,\}, \quad \mathcal{A}'_p = \{\,(a,b) \in \mathcal{A} \mid a = b \;\&\; p \mid \gcd(a,b)\,\},$ wobei p eine Primzahl ist,
- $\mathcal{B}_p = \{\,(a,b) \in \mathcal{B} \mid a \ne b \;\&\; p \mid \gcd(a,b)\,\}, \quad \mathcal{B}'_p = \{\,(a,b) \in \mathcal{B} \mid a = b \;\&\; p \mid \gcd(a,b)\,\},$ wobei p eine Primzahl ist,

Die folgenden Aussagen sind leicht einzusehen:

1. Es gilt $\gcd(a,b) \le d$ für alle $(a,b) \in \mathcal{A}$ und alle $(a,b) \in \mathcal{B}$ mit $a \ne b$.
2. Für alle Primzahlen p gilt die Ungleichung $\operatorname{card} \mathcal{B}_p \le \operatorname{card} \mathcal{A}_p$.
3. Es gilt $\operatorname{card} \mathcal{A} = \operatorname{card} \mathcal{B}$.
4. Es gilt $\mathcal{A} = \mathcal{A}' + \bigcup_{p \text{ prim}} \mathcal{A}_p + \bigcup_{p \text{ prim}} \mathcal{A}'_p$, woraus folgt
$$\operatorname{card} \mathcal{A}' = \operatorname{card} \mathcal{A} - \sum_{p \text{ prim}} \operatorname{card} \mathcal{A}_p - d$$
5. Es gilt $\mathcal{B} = \mathcal{B}' + \bigcup_{p \text{ prim}} \mathcal{B}_p + \bigcup_{p \text{ prim}} \mathcal{B}'_p$, woraus folgt
$$\operatorname{card} \mathcal{B} - \sum_{p \text{ prim}} \operatorname{card} \mathcal{B}_p - (d+1) \le \operatorname{card} \mathcal{B}'$$

Die Aussage des Lemmas folgt unmittelbar aus der folgenden Ungleichung

$$\begin{aligned} \operatorname{card} \mathcal{A}' &= \operatorname{card} \mathcal{A} - \sum_{p \text{ prim}} \operatorname{card} \mathcal{A}_p - d = \operatorname{card} \mathcal{B} - \sum_{p \text{ prim}} \operatorname{card} \mathcal{A}_p - d \\ &\le \operatorname{card} \mathcal{B} - \sum_{p \text{ prim}} \operatorname{card} \mathcal{B}_p - d \le \operatorname{card} \mathcal{B}' + 1. \end{aligned}$$

□

Aus Satz 2.2.1 und Lemma 2.2.2 folgt unmittelbar

Satz 2.2.3 *Seien $c \in \mathbb{Z}$ und $d \in \mathbb{N}$. Ist $d \ge 2$, dann gilt*

$$\frac{\operatorname{card}\{\,(a,b) \in \mathbb{Z}^2 \mid c \le a,b < c+d \text{ und } \gcd(a,b) = 1\,\}}{\operatorname{card}\{\,(a,b) \in \mathbb{Z}^2 \mid c \le a,b < c+d\,\}} \ge \frac{1}{4}.$$

Ist $d \ge 10$, dann gilt

$$\frac{\operatorname{card}\{\,(a,b) \in \mathbb{Z}^2 \mid c \le a,b < c+d \text{ und } \gcd(a,b) = 1\,\}}{\operatorname{card}\{\,(a,b) \in \mathbb{Z}^2 \mid c \le a,b < c+d\,\}} \ge \frac{1}{2}.$$

Beweis. Wir verwenden die gleichen Bezeichnungen wie im Beweis von Lemma 2.2.3. Außerdem seien $\mathcal{C} = \{\,(a,b) \in \mathbb{Z}^2 \mid 1 \leq a,b < d\,\}$ und $\mathcal{C}' = \{\,(a,b) \in \mathbb{Z}^2 \mid 1 \leq a,b < d \text{ und } \gcd(a,b) = 1\,\}$. Mit $d > 2$ gilt

$$\frac{\operatorname{card}\mathcal{B}'}{\operatorname{card}\mathcal{B}} \geq \frac{\operatorname{card}\mathcal{A}' - 1}{\operatorname{card}\mathcal{A}} \geq \frac{\operatorname{card}\mathcal{C}'}{\operatorname{card}\mathcal{C}} \frac{\operatorname{card}\mathcal{C}}{\operatorname{card}\mathcal{A}} \geq \frac{6}{\pi^2}\frac{4}{9} \geq \frac{1}{4}.$$

Analog gilt für $d > 10$

$$\frac{\operatorname{card}\mathcal{B}'}{\operatorname{card}\mathcal{B}} \geq \frac{\operatorname{card}\mathcal{A}' - 1}{\operatorname{card}\mathcal{A}} \geq \frac{\operatorname{card}\mathcal{C}'}{\operatorname{card}\mathcal{C}} \frac{\operatorname{card}\mathcal{C}}{\operatorname{card}\mathcal{A}} \geq \frac{6}{\pi^2}\frac{100}{121} \geq \frac{1}{2}.$$

□

2.3 Kettenbruchentwicklung

In diesem Abschnitt präsentieren wir die Definition eines Kettenbruchs sowie einen Satz über die Kettenbruchentwicklung. Mehr zu Kettenbrüchen kann in [Per57] nachgelesen werden.

Kettenbruche stellen jede beliebige reele Zahl eindeutig in der folgenden Form dar:

$$a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cdots}}},$$

wobei a_0 eine ganze und $a_1, a_2, a_3, \ldots$ natürliche Zahlen sind. Der Algorithmus zur Bestimmung des Kettenbruchs kann z.B. in [NC00] gefunden werden.

Es gilt der folgende Satz.

Satz 2.3.1 (Kettenbruchentwicklung) *Seien $a, b, c, d \in \mathbb{N}$, so daß*

$$\left|\frac{a}{b} - \frac{c}{d}\right| < \frac{1}{2b^2}$$

gilt, dann berechnet der Algorithmus zur Kettenbruchentwicklung bei Eingabe von c und d die Zahlen a und b.

Beweis. Siehe [NC00], Satz 5.1, Seite 229.

2.4 Gitter

In diesem Abschnitt präsentieren wir die grundlegenden Definitionen und Sätze aus der Gittertheorie. Weitere Details können in [MG02] und [Sch] nachgelesen werden.

2.4.1 Grundlegende Definitionen

Satz 2.4.1 (Gitter) *Seien $n \in \mathbb{N}$ und $\mathbf{b}_1, \ldots, \mathbf{b}_k \in \mathbb{R}^n$ linear unabhängige Vektoren. Die Menge*

$$L = \{\, \lambda \in \mathbb{R}^n \mid \lambda = \sum_{i=1}^{k} \mathbf{b}_i \mathbb{Z} \,\}$$

heißt Gitter. Die Folge $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ heißt Basis des Gitters L. Die Zahl k heißt Dimension von L. Um zu verdeutlichen, daß das Gitter L von $\mathsf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_k)$ erzeugt wird, schreiben wir $L = L(\mathsf{B})$. Wenn $n = k$ ist, dann ist das Gitter L vollständig.

Satz 2.4.2 *Sei $\mathcal{M} \subset \mathbb{R}^n$. Wir bezeichnen mit*

$$\operatorname{span} \mathcal{M} = \{ \sum_{i=1}^{k} \mathbf{m}_i \mathbb{R} \mid \mathbf{m}_i \in \mathcal{M},\ k \in \mathbb{N} \},$$

den von $\mathcal{M}$ aufgespannten linearen Vektorraum.

Satz 2.4.3 (Determinante eines Gitters) *Sei $\mathsf{B} \in \mathbb{R}^{n \times k}$ und $L = L(\mathsf{B}) \subset \mathbb{R}^n$ ein Gitter. Dann ist die Determinante von L wie folgt definiert*

$$\det L = \sqrt{\det(\mathsf{B}^T \mathsf{B})}.$$

Geometrisch gesehen ist die Determinante das n-dimensionale Volumen einer Grundmasche $\{\, \mathbf{x} \in \mathbb{R}^n \mid \mathbf{x} = \sum_{i=1}^{k} a_i \mathbf{b}_i,\ 0 \le a_1, \ldots, a_k < 1 \,\}$

Satz 2.4.4 (inneres Produkt) *Seien $\mathbf{x} = (x_1, \ldots, x_n), \mathbf{y} = (y_1, \ldots, y_n) \in \mathbb{R}^n$ zwei Vektoren. Das innere Produkt von $\mathbf{x}$ und $\mathbf{y}$ ist*

$$\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x} \cdot \mathbf{y} = \sum_{i=1}^{k} x_i y_i.$$

Satz 2.4.5 (duales Gitter) *Sei $L \subset \mathbb{R}^n$ ein Gitter. Das zu L duale Gitter ist*

$$L^* = \{\, \mathbf{x} \in \operatorname{span}(L) \mid \langle \mathbf{x}, \mathbf{b} \rangle \in \mathbb{Z} \text{ für alle } \mathbf{b} \in L \,\}.$$

Satz 2.4.6 *Folgende Aussagen sind wahr:*

- $\det L^* = 1/\det L$,
- $(L^*)^* = L$,
- *falls $L = L(\mathsf{B})$ ein vollständiges Gitter ist, dann gilt $L^* = L(\mathsf{B}^{-1})$.*

Beweis. Siehe [Sch].

Satz 2.4.7 (Normen) *Sei $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$ ein Vektor. Dann setze*

- $\|\mathbf{x}\|_1 = |x_1| + |x_2| + \cdots + |x_n|$,

- $\|\mathbf{x}\|_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$ *und*
- $\|\mathbf{x}\|_\infty = \max\{|x_1|, \ldots, |x_n|\}$.

Sei $\mathsf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_k) \in \mathbb{R}^{n \times k}$ *eine Matrix. Dann setze*

1. $\|\mathsf{B}\|_1 = \max_j \sum_{i=1}^n |b_{ij}|$ *und*
2. $\|\mathsf{B}\|_\infty = \max_i \sum_{j=1}^k |b_{ij}|$.

Satz 2.4.8 *Sei* $\mathbf{x} \in \mathbb{R}^n$ *ein Vektor, dann gilt*

1. $\|\mathbf{x}\|_2 \leq \|\mathbf{x}\|_1 \leq \sqrt{n}\|\mathbf{x}\|_2$ *und*
2. $\|\mathbf{x}\|_\infty \leq \|\mathbf{x}\|_2 \leq \sqrt{n}\|\mathbf{x}\|_\infty$.

Satz 2.4.9 (sukzessive Minima) *Sei* $||\cdot||$ *eine beliebige Norm. Die sukzessive Minima* $\lambda_1, \ldots, \lambda_k$ *eines Gitters* $L \subset \mathbb{R}^n$ *vom Rang* k *bezüglich der Norm* $||\cdot||$ *sind*

$$\lambda_i = \lambda_i(L) = \inf\{\, r \in \mathbb{R} \mid r > 0 \text{ und es gibt } i \text{ linear unabhängige Vektoren } \mathbf{v}_1, \ldots, \mathbf{v}_i \in L \text{ mit } ||\mathbf{v}_j|| \leq r \text{ für } 1 \leq j \leq i \,\}.$$

Sukzessive Minima sind wichtige geometrische Invarianten eines Gitters.

Satz 2.4.10 *Sei* $||\cdot||$ *eine beliebige Norm. Sei außerdem* $L \subset \mathbb{R}^n$ *ein Gitter vom Rang* k*. Wir setzen*

$$\nu(L) = \inf\{\, r \in \mathbb{R} \mid r > 0 \text{ und es gibt eine Basis } (\mathbf{b}_1, \ldots, \mathbf{b}_k) \text{ von } L, \text{ so daß } ||\mathbf{b}_j|| \leq r \text{ für } 1 \leq j \leq k \,\}.$$

Satz 2.4.11 (Transferenz Theorem) *Sei* $L \subset \mathbb{R}^n$ *ein Gitter, dann gilt*

$$1 \leq \lambda_1(L)\nu(L^*) \leq n.$$

Beweis. Siehe [Ban93].

2.4.2 Hermite- und Smith-Normalformen

Satz 2.4.12 (Hermite-Normalform) *Sei* $n \in \mathbb{N}$*. Die Matrix* $\mathsf{M} = (m_{i,j})_{1 \leq i,j \leq n} \in \mathbb{Z}^{n \times n}$ *ist in* Hermite-Normalform *(kurz* HNF*), wenn gilt:*

- $m_{i,j} = 0$ *für alle* $i > j$,
- $m_{i,i} > 0$ *für* $i = j$, $1 \leq i, j \leq n$ *und*
- $m_{i,j} < m_{i,i}$ *für* $1 \leq i < j < n$.

Satz 2.4.13 (Eindeutigkeit der HNF) *Seien $n \in \mathbb{N}$ und $\mathsf{M} = (m_{i,j})_{1 \le i,j \le n} \in \mathbb{Z}^{n \times n}$ eine Matrix mit $\det \mathsf{M} \neq 0$, dann existiert genau eine Matrix B in Hermite-Normalform mit $\mathsf{B} = \mathsf{MU}$, wobei $\mathsf{U} \in GL(n, \mathbb{Z})$ ist, d.h. $\mathsf{U} \in \mathbb{Z}^{n \times n}$ und $\det(\mathsf{U}) = \pm 1$.*

Satz 2.4.14 (Berechnung der HNF) *Seien $k, T \in \mathbb{N}$ und $\mathsf{M} = (m_{i,j}) \in \mathbb{Z}^{n \times n}$ eine Matrix mit $\det \mathsf{M} \neq 0$ und $|m_{i,j}| \le T$ für alle $1 \le i, j \le n$. Dann existiert ein deterministischer Algorithmus, der bei Eingabe der Matrix M und einer positiven Zahl $h = k \det \mathsf{M}$ die Hermite-Normalform von M berechnet. Die Laufzeit des Algorithmus ist $O(n^2 B(\log T) + n^3 B(\log h))$, wobei $B(t) = t \log t \log \log t$ ist.*

Beweis. Siehe [HM91].

Satz 2.4.15 (Smith-Normalform) *Sei $n \in \mathbb{N}$. Die Matrix $\mathsf{M} = (m_{i,j})_{1 \le i,j \le n} \in \mathbb{Z}^{n \times n}$ ist in* Smith-Normalform*, wenn gilt:*

- $m_{i,j} = 0$ *für alle* $i \neq j$, $1 \le i, j \le n$,
- $m_{i,i} > 0$ *für* $1 \le i \le n$,
- $m_{i+1,i+1} | m_{i,i}$ *für* $1 \le i < n$.

Satz 2.4.16 (Eindeutigkeit der Smith-Normalform) *Sei $\mathsf{M} \in \mathbb{Z}^{n \times n}$ eine Matrix mit $\det \mathsf{M} \neq 0$. Dann existiert genau eine Matrix B in Smith-Normalform und invertierbare Matrizen $\mathsf{U}, \mathsf{V} \in GL(n, \mathbb{Z})$, so daß $\mathsf{B} = \mathsf{UMV}$ gilt.*

Satz 2.4.17 (Berechnung der Smith-Normalform) *Seien $k, T \in \mathbb{N}$ und $\mathsf{M} = (m_{i,j}) \in \mathbb{Z}^{n \times n}$ eine Matrix mit $\det \mathsf{M} \neq 0$ und $|m_{i,j}| \le T$ für alle $1 \le i, j \le n$. Es existiert ein deterministischer Algorithmus* SNF*, der bei Eingabe von M und einer positiven Zahl $h = k \det \mathsf{M}$ die Smith-Normalform von M berechnet. Die Laufzeit des Algorithmus ist $O(n^2 B(\log T) + n^3 B(\log h) \log h)$, wobei $B(t) = t \log t \log \log t$ ist.*

Beweis. Siehe [HM91].

2.4.3 Reduzierte Basen

Satz 2.4.18 (Gram-Schmidt-Zerlegung) *Sei $\mathsf{B} \in \mathbb{R}^{n \times k}$ eine Gitterbasis. Die Zerlegung $\mathsf{B} = \hat{\mathsf{B}}\mathsf{R}$ in eine Matrix $\hat{\mathsf{B}} \in \mathbb{R}^{n \times k}$ mit paarweise orthogonalen Spalten und eine obere Dreiecksmatrix $\mathsf{R} \in \mathbb{R}^{k \times k}$ mit $\det(\mathsf{R}) = 1$ nennen wir die* Gram-Schmidt-Zerlegung.

Ein Algorithus zur Bestimmung der Gramm-Schmidt-Zerlegung ist in [Coh00] angegeben.

Satz 2.4.19 (LLL-reduzierte Basis) *Seien $\mathsf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_k) \in \mathbb{Z}^{n \times k}$ eine Gitterbasis und $\mathsf{B} = \hat{\mathsf{B}}\mathsf{R}$, mit $\hat{\mathsf{B}} = (\hat{\mathbf{b}}_1, \ldots, \hat{\mathbf{b}}_k)$ und $\mathsf{R} = (r_{i,j})$, die Gram-Schmidt-Zerlegung von B. Dann heißt B* LLL-reduziert *genau dann, wenn*

$$|r_{i,j}| \le \frac{1}{2} \qquad \text{für } 1 \le i < j \le n \text{ und}$$

$$||\hat{\mathbf{b}}_{j+1}||^2 \ge (\frac{3}{4} - r_{j,j+1}^2)||\hat{\mathbf{b}}_j||^2 \qquad \text{für } 1 \le j < n.$$

Satz 2.4.20 *Sei $L \subset \mathbb{R}^{n\times k}$ ein Gitter mit Basis $\mathsf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_k)$. Sei außerdem $\mathsf{B} = \hat{\mathsf{B}}\mathsf{R}$, mit $\hat{\mathsf{B}} = (\hat{\mathbf{b}}_1, \ldots, \hat{\mathbf{b}}_k)$ und $\mathsf{R} = (r_{i,j})$, die Gram-Schmidt-Zerlegung von B. Für $j \in \{1, \ldots, n\}$ setze*

$$\mathsf{P}_j = (\hat{\mathbf{b}}_1/||\hat{\mathbf{b}}_1||, \ldots, \hat{\mathbf{b}}_{j-1}/||\hat{\mathbf{b}}_{j-1}||) \in \mathbb{R}^{n\times(j-1)}.$$

Die Abbildung

$$\pi_j : \mathbb{R}^n \longrightarrow \mathbb{R}^n : \mathbf{x} \longmapsto \mathbf{x} - \mathsf{P}_j\mathsf{P}^T\mathbf{x}$$

ist eine orthogonale Projektion von $\mathbb{R}^n$ auf $\mathcal{W} := \{\mathbf{b}_1, \ldots, \mathbf{b}_{j-1}\}^\perp$.

Eine Projektion von L auf $\mathcal{W}_j$ bezeichnen wir mit $L_j(\mathsf{B}) := \{\, \pi_j(\mathbf{v}) \mid \mathbf{v} \in L \,\}$.

Satz 2.4.21 (Korkine-Zolotarev-reduzierte Basis) *Wir nennen eine Gitterbasis $\mathsf{B} \in \mathbb{R}^{n\times k}$ mit Gram-Schmidt-Zerlegung $\mathsf{B} = \hat{\mathsf{B}}\mathsf{R}$, $\hat{\mathsf{B}} = (\hat{\mathbf{b}}_1, \ldots, \hat{\mathbf{b}}_k)$, $\mathsf{R} = (r_{i,j})$, genau dann Korkine-Zolotarev-reduziert, wenn gilt*

- *$|r_{i,j}| \leq 1/2$, für alle $1 \leq i < j \leq k$, und*
- *$||\hat{\mathbf{b}}_j|| = \lambda_1(L_j(\mathsf{B}))$, für alle $1 \leq j \leq k$.*

In der Dissertation von Christoph Ludwig (siehe [Lud05]) werden weitere Definitionen von reduzierten Basen, sowie effiziente Algorithmen zu deren Bestimmung präsentiert und untersucht.

2.4.4 Periodengitter einer Funktion

Satz 2.4.22 *Seien $n \in \mathbb{N}$, $\mathcal{S}$ eine Menge, $f : \mathbb{R}^n \longrightarrow \mathcal{S}$ eine Funktion und $L \subset \mathbb{R}^n$ ein Gitter. Die Funktion f ist genau dann periodisch mit Periodengitter $L = L_f$, wenn für alle $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ gilt:*

$$f(\mathbf{x}) = f(\mathbf{y}) \Leftrightarrow \mathbf{x} - \mathbf{y} \in L.$$

2.4.5 Wahrscheinlichkeit für die Erzeugung

Als nächstes stellen wir fest, wie viele Vektoren aus einem Gitter M gebraucht werden, damit sie mit hoher Wahrscheinlichkeit das ganze Gitter M erzeugen. Dazu beweisen wir den folgenden Satz.

Satz 2.4.23 *Seien $r, n \in \mathbb{N}$, $n > 2^{2r}r^{r/2-1/2}$, $M \subset \mathbb{R}^r$ ein vollständiges Gitter und $\mathcal{B} = \{\, \mathbf{x} \in \mathbb{R}^r \mid 0 \leq x_i < n\nu(M),\ 1 \leq i \leq r \,\}$ eine Teilmenge von $\mathbb{R}^r$. Sei außerdem*

$$\mathcal{A} = \{\, (\mathbf{x}_1, \ldots, \mathbf{x}_{2r}) \in (M \cap \mathcal{B})^{2r} \mid \dim \operatorname{span}(\mathbf{x}_1, \ldots, \mathbf{x}_r) = r \text{ und } \langle \mathbf{x}_1, \ldots, \mathbf{x}_{2r} \rangle = M \,\},$$

dann gilt

$$\frac{\operatorname{card} \mathcal{A}}{(\operatorname{card} M \cap \mathcal{B})^{2r}} \geq \frac{1}{2^{4r+2}}.$$

Beweis. Wir führen den Beweis in zwei Schritten. Als erstes schätzen wir den Anteil der Vektoren $\mathbf{x}_1, \ldots, \mathbf{x}_r$, die einen Vektorraum der Dimension r erzeugen, von unten ab. Als zweites geben wir die untere Schranke für den Anteil der Vektoren $\mathbf{x}_{r+1}, \ldots, \mathbf{x}_{2r}$, die zusammen mit den Vektoren aus dem ersten Schritt das komplette Gitter M erzeugen.

Seien $\mathbf{x}_1, \ldots, \mathbf{x}_r \in \mathbb{R}^r$ linear unabhängige Vektoren. Für die Teilgitter

$$M_i = M \cap \operatorname{span}(\mathbf{x}_1, \ldots, \mathbf{x}_i), \quad 1 \le i \le r,$$

gilt $\det M_i \le \det M_{i-1}\nu(M)$. Daraus folgt

$$\begin{aligned}
&\frac{\operatorname{card} M_{i-1} \cap \mathcal{B}}{\operatorname{card} \mathcal{M}_i \cap \mathcal{B}} \le \frac{(\sqrt{r}n\nu(M) + 2\nu(M))^{i-1}}{\det M_{i-1}} \frac{\det M_i}{(n\nu(M) - 2\nu(M))^i} \\
\le &\frac{\nu(M)}{n\nu(M) - 2\nu(M)} \frac{(\sqrt{r}n\nu(M) + 2\nu(M))^{i-1}}{(n\nu(M) - 2\nu(M))^{i-1}} = \frac{1}{n-2}\left(\frac{\sqrt{r}n+2}{n-2}\right)^{i-1} \\
\le &\frac{2}{n}\left(\frac{2\sqrt{r}n}{n/2}\right)^{i-1} \le \frac{2(4\sqrt{r})^{i-1}}{2^{2r}r^{r/2-1/2}} \le \frac{2(4\sqrt{r})^{r-1}}{2^{2r}r^{r/2-1/2}} \le \frac{1}{2}.
\end{aligned}$$

Daraus leiten wir ab, daß

$$1 \le \frac{1}{2}\operatorname{card} M_1 \cap \mathcal{B} \le \frac{1}{4}\operatorname{card} M_2 \cap \mathcal{B} \le \ldots \le \frac{1}{2^r}\operatorname{card} M_r \cap \mathcal{B} \tag{2.1}$$

Wir betrachten den folgenden Prozeß: Wir ziehen nacheinander r Vektoren aus $M \cap \mathcal{B}$ und schätzen die Wahrscheinlichkeit von unten ab, daß der jeweils neue Vektor nicht im Untervektorraum liegt, der von bereits gezogenen Vektoren erzeugt wird. Aus (2.1) folgt, daß

$$\begin{aligned}
&\frac{\operatorname{card}\{\, (\mathbf{x}_1, \ldots, \mathbf{x}_r) \in (M \cap \mathcal{B})^r \mid \dim \operatorname{span}(\mathbf{x}_1, \ldots, \mathbf{x}_r) = r \,\}}{(\operatorname{card} M \cap \mathcal{B})^r} \ge \\
&\left(1 - \frac{1}{2^r}\right)\left(1 - \frac{1}{2^{r-1}}\right)\cdots\left(1 - \frac{1}{2}\right) > \prod_{i=1}^{\infty}\left(1 - \frac{1}{2^i}\right) > \frac{1}{4}.
\end{aligned}$$

Die letzte Ungleichung folgt aus Eulers Pentagonalzahlen-Theorem:

$$\prod_{i=1}^{\infty}(1 - x^i) = \sum_{i=-\infty}^{\infty}(-1)^i x^{i(3i+1)/2}.$$

Als nächstes berechnen wir die untere Schranke dafür, daß die Vektoren $\mathbf{x}_1, \ldots, \mathbf{x}_{2r}$ das komplette Gitter M erzeugen. Sei $\mathsf{C} = (\mathbf{c}_1, \ldots, \mathbf{c}_r)$ eine Korkin-Zolotarev-reduzierte Basis von M. Für $i = 1, \ldots, r$ sei

$$\pi_i : M \longrightarrow \mathbb{Z}\mathbf{c}_i$$

die Projektion auf die Basis C und

$$M_i = M \cap \operatorname{span}(\mathbf{x}_1, \mathbf{x}_{r+1}, \ldots, \mathbf{x}_i, \mathbf{x}_{r+i}).$$

Wir gehen iterativ vor und schätzen für $i = 1, \ldots, r$ die untere Schranke für die Wahrscheinlichkeit ab, daß das Untergitter M_i Vektoren $\mathbf{u}_1, \ldots, \mathbf{u}_i$ enthält, für die gilt

$$\pi_i(\mathbf{u}_j) = \begin{cases} 1, & \text{wenn } i - j = 0 \\ 0, & \text{wenn } i - j < 0 \end{cases}. \tag{2.2}$$

Es ist leicht einzusehen, daß die Vektoren $\mathbf{u}_1, \ldots, \mathbf{u}_r$, welche die Gleichung (2.2) erfüllen, das komplette Gitter M erzeugen.

Sei $i \in \{1, \ldots, r\}$ beliebig und dann fest. Wir betrachten zwei Vektoren $\mathbf{x}_i, \mathbf{x}_{r+i} \in M \cap \mathcal{B}$ mit $\mathbf{x}_j = \mathbf{w}_j + t_j \mathbf{c}_r$, $\mathbf{w}_j \in M$ und $\pi_i(\mathbf{w}_j) = \mathbf{0}$ für $j = i, r+i$. Das durch $\mathbf{u}_1, \ldots, \mathbf{u}_{i-1}, \mathbf{x}_i, \mathbf{x}_{r+i}$ erzeugte Untergitter von M enthält genau dann einen Vektor $\mathbf{u}_i$, welcher (2.2) erfüllt, wenn $\gcd(A_1 + t_1, A_2 + t_2) = 1$ mit $A_1, A_2 \in \mathbb{Z}$ gilt. Dabei hängen A_1 und A_2 von Vektoren $\mathbf{u}_1, \ldots, \mathbf{u}_{i-1}, \mathbf{w}_i$, und $\mathbf{w}_{r+i}$ ab. Die Anzahl der teilerfremden Paare $(A_1 + t_1, A_2 + t_2)$ ist mindestens $1/16$. Dies folgt aus dem Satz 2.2.3, wenn $A_1 + t_1, A_2 + t_2 \in \{c, \ldots, c + d\}$ mit $d \geq 2$ gilt. Diese Bedingung ist durch die Wahl von $\mathcal{B}$ erfüllt. Deshalb gilt

$$\frac{\operatorname{card} \mathcal{A}}{(\operatorname{card} M \cap \mathcal{B})^{2r}} \geq \frac{1}{4} \left(\frac{1}{16} \right)^r = \frac{1}{2^{4r+2}}.$$

□

2.4.6 Approximationsschranken zum Invertieren

Mit dem Satz in diesem Abschnitt beweisen wir, daß ein Polynomzeit-Algorithmus existiert, welcher bei der Eingabe des Erzeugendensystems eines Gitters M das duale Gitter M^* berechnet.

Satz 2.4.24 *Sei $M \subset \mathbb{R}^r$ ein vollständiges Gitter und $\mathbf{a}_1, \ldots, \mathbf{a}_{2r} \in M$ Vektoren, die M erzeugen. Nehme an, daß $\mathbf{a}_1, \ldots, \mathbf{a}_r$ linear unabhängig sind und ein Untergitter M_r erzeugen. Wähle*

$$\alpha \geq \max\{\|\mathbf{a}_j\|_2 : 1 \leq j \leq 2r\}, \quad \mu \leq \lambda_1(M) \quad \textit{und} \quad \epsilon > 0. \tag{2.3}$$

Seien $D_{min} < \det(M)$ und $\lambda_{max} > \lambda_r(M_r)$. Seien außerdem $\mathbf{a'}_1, \ldots, \mathbf{a'}_{2r} \in \mathbb{Z}^r$ Vektoren mit

$$\|\mathbf{a'}_j - 2^h \mathbf{a}_j\|_2 \leq \frac{\sqrt{r}}{2}, \quad 1 \leq j \leq 2r, \tag{2.4}$$

und

$$h > \log_2 \frac{2^{r^2+3r+4} \alpha^r r^{r+4} \lambda_{max}^r (2^{r^2-1} r^r \lambda_{max}^{r-1} / D_{min} + \epsilon)}{\epsilon \mu D_{min}^2}. \tag{2.5}$$

Dann existiert ein klassischer Polynomzeit-Algorithmus, welcher bei Eingabe $\mathbf{a'}_1, \ldots, \mathbf{a'}_{2r}$ Vektoren $\mathbf{c'}_1, \ldots, \mathbf{c'}_r \in \mathbb{Q}^r$ berechnet, welche eine Basis $\mathsf{C} = (\mathbf{c}_1, \ldots, \mathbf{c}_r)$ des dualen Gitters M^* approximieren, so daß gilt:

$$\|\mathbf{c'}_j - \mathbf{c}_j\|_1 < \epsilon, \quad 1 \leq j \leq r. \tag{2.6}$$

Die Laufzeit dieses Algorithmus ist $O(r^6 \log^3(2^h \max\{\alpha, r^{\frac{3}{2}} \lambda_r(M_r)\}))$.

Beweis. Es gelten alle Definitionen und Annahmen wie im Satz.

Die Idee, auf der der Algorithmus beruht, ist die folgende: Als erstes wenden wir den Algorithmus von Buchmann und Pohst (siehe [BP89] und [BK93]) auf die Eingabevektoren $\mathbf{a'}_1, \ldots, \mathbf{a'}_{2r}$ an und berechnen damit eine Approximation einer Basis $\mathsf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_r)$ von M. Sei $\mathsf{B'} = (\mathbf{b'}_1, \ldots, \mathbf{b'}_r)$ diese Approximation. Als nächstes invertieren wir die Matrix $\mathsf{B'}$ und erhalten die Matrix $\mathsf{C'} = (\mathbf{c'}_1, \ldots, \mathbf{c'}_r)$, welche die Basis C des dualen Gitters M^* wie in (2.6) approximiert.

Seien γ, p, λ, q Konstanten mit folgenden Eigenschaften:

1. $\gamma = 2^r \lambda_r(M_r)(2r + \sqrt{2})$,
2. $p = \left\lceil \log_2 \left(\frac{2^{(r^2)} r^{5/2} \lambda_{max}^r (2r+\sqrt{2})^{r-1}}{\epsilon D_{min}} \left(\frac{2^{r(r-1)} \lambda_{max}^{r-1} r (2r+\sqrt{2})^{r-1}}{D_{min}} + \epsilon \right) \right) \right\rceil$, so daß gilt $p > \log_2 \left(\frac{2^r \gamma^{r-1} r^{5/2} \lambda_r(M_r)}{\epsilon \det(M)} \left(\frac{r\gamma^{r-1}}{\det(M)} + \epsilon \right) \right)$,
3. $\lambda := \frac{\sqrt{r}\alpha^r}{\det(M)}(r(1 + 2^{-p}) + \sqrt{2})$
4. $q = h - p$. Dabei gilt $q > \log_2 \left(\frac{2^{r-1}\lambda}{\mu}(r(1 + 2^{-p}) + \sqrt{2}) \right)$.

Wir beschreiben nun den Algorithmus genauer. Zuerst berechnen wir Vektoren $\hat{\mathbf{a}}_1, \ldots, \hat{\mathbf{a}}_{2r} \in \mathbb{Z}^r$ mit

$$\hat{a}_{i,j} = \lfloor 2^{-p} a'_{i,j} \rceil, \quad 1 \le i \le r,\ 1 \le j \le 2r,$$

wobei $\lfloor x \rceil$ die zu $x \in \mathbb{R}$ nächste ganze Zahl ist. Diese Vektoren sind kürzer als die Ausgangsvektoren $\mathbf{a'}_i$, $1 \le i \le 2r$, was einen Vorteil hat, da wir dadurch im weiteren eine Matrix mit kleineren Einträgen bekommen. Auch diese Vektoren approximieren die Vektoren $\mathbf{a}_1, \ldots, \mathbf{a}_{2r} \in \mathbb{R}^r$. Es gilt nämlich

$$\|\hat{\mathbf{a}}_j - 2^q \mathbf{a}_j\|_2 \le \frac{\sqrt{r}}{2}(1 + 2^{-p}), \quad 1 \le j \le 2r. \tag{2.7}$$

Wie in [BK93] definieren wir ein neues Gitter $A \subset \mathbb{Z}^{(4r)\times(2r)}$, dessen Basis durch die Spaltenvektoren der Matrix $\tilde{\mathsf{A}} = (\tilde{\mathbf{a}}_1, \ldots, \tilde{\mathbf{a}}_{2r})$ gegeben ist. Diese Vektoren haben die Form $\tilde{\mathbf{a}}_i = (\mathbf{e}_i, \hat{\mathbf{a}}_i)$, $1 \le i \le 2r$, wobei $\mathbf{e}_i$ der i-te Einheitsvektor der Dimension $2r$ ist. Wir reduzieren die Basis $\tilde{\mathsf{A}}$ mit dem LLL-Algorithmus (siehe [LLL82]) und bekommen die LLL-reduzierte Basis $\tilde{\mathsf{B}} = (\tilde{\mathbf{b}}_1, \ldots, \tilde{\mathbf{b}}_{2r})$ mit $\tilde{\mathbf{b}}_j = (\mathbf{m}_j, \hat{\mathbf{b}}_j)$, $1 \le j \le 2r$. Daraus extrahieren wir die Matrix $(\mathbf{m}_{r+1}, \ldots, \mathbf{m}_{2r})$ und berechnen

$$\mathsf{B'} = (\mathbf{b'}_1, \ldots, \mathbf{b'}_r) = (\mathbf{a'}_1, \ldots, \mathbf{a'}_{2r})(\mathbf{m}_{r+1}, \ldots, \mathbf{m}_{2r}).$$

Schließlich invertieren wir die Matrix $\mathsf{B'}$ und erhalten die Matrix $\mathsf{C'}$, die eine Basis von M^* approximiert.

Wir zeigen nun, daß die Approximationsschranken aus dem Satz wahr sind. Es lassen sich folgende Aussagen beweisen:

1. $\|\mathbf{m}_j\|_2 \le 2^{r+q+1/2} \lambda_{j-r}(M_r)$, $r + 1 \le j \le 2r$.
2. $\mathsf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_r) = (\mathbf{a}_1, \ldots, \mathbf{a}_{2r})(\mathbf{m}_{r+1}, \ldots, \mathbf{m}_{2r})$ ist eine Basis des Gitters M und es gilt $\|\mathbf{b}_j\|_2 \le \gamma$,
3. $\mathsf{B'} = (\mathbf{b'}_1, \ldots, \mathbf{b'}_r) = (\mathbf{a'}_1, \ldots, \mathbf{a'}_{2r})(\mathbf{m}_{r+1}, \ldots, \mathbf{m}_{2r})$ ist eine Approximation von B und es gilt $\|2^{-p-q}\mathbf{b'}_j - \mathbf{b}_j\|_2 \le 2^{r-p} r \lambda_j(M_r)$, $1 \le j \le r$.

Die Beweise lassen sich, bis auf die Definitionen von λ und q, eins zu eins aus [BK93] übertragen. λ und q müssen deshalb geändert werden, weil die Approximationsschranke (2.7) größer als in [BK93] ist.

Sei $\mathsf{C} = \mathsf{B}^{-1} = (\mathbf{c}_1, \ldots, \mathbf{c}_r)$. Wir bestimmen die obere Schranke für $|c_{i,j}|$, $1 \leq i, j \leq r$. Sie folgt aus $\sum_{i=1}^{r} c_{i,j}\mathbf{b}_i = \mathbf{e}_j$, der Cramerscher Regel und der Hadamard-Ungleichung. Es gilt also

$$|c_{i,j}| = |\frac{\det(\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}, \mathbf{e}_j, \mathbf{b}_{i+1}, \ldots, \mathbf{b}_r)}{\det(\mathbf{b}_1, \ldots, \mathbf{b}_r)}| \leq \frac{\gamma^{r-1}}{\det(B)}. \tag{2.8}$$

Wir fassen noch einmal das zusammen, was wir bis jetzt explizit oder implizit berechnet haben. Wir haben

1. eine Matrix $\mathsf{B} \in \mathbb{R}^{r \times r}$, deren Spalten eine Basis von M bilden. Diese Matrix ist implizit gegeben und es gilt
$$||\mathsf{B}||_1 \leq \sqrt{r}\gamma,$$

2. eine Matrix $\mathsf{C} = \mathsf{B}^{-1} \in \mathbb{R}^{r \times r}$, die auch nur implizit gegeben ist. Es gilt
$$\|\mathsf{C}\|_1 \leq \frac{r\gamma^{r-1}}{\det(B)},$$

3. eine Matrix $\mathsf{B}' = \mathsf{B} + \mathsf{F} \in \mathbb{Q}^{r \times r}$, welche wir explizit ausgerechnet haben. Es gilt
$$\|\mathsf{F}\|_1 \leq 2^{r-p} r\sqrt{2r+1}\lambda_r(M_r).$$

Wir schätzen nun den Fehler $\|\mathsf{C}' - \mathsf{C}\|_1$ ab, welcher besagt, wie groß der Abstand zwischen der Basis C von M^* und der berechneten Approximation C' ist. Aus der numerischen Mathematik kennen wir folgende Ungleichung:

$$\|(\mathsf{B}+\mathsf{F})^{-1} - \mathsf{B}^{-1}\|_1 \leq \frac{\|\mathsf{B}^{-1}\|_1^2 \cdot \|\mathsf{F}\|_1}{1 - \|\mathsf{B}^{-1}\|_1 \cdot \|\mathsf{F}\|_1}. \tag{2.9}$$

Wir setzen die Zahlen, die wir oben ausgerechnet haben in (2.9) ein und erhalten

$$\|\mathsf{C}' - \mathsf{C}\|_1 = \|(\mathsf{B}+\mathsf{F})^{-1} - \mathsf{B}^{-1}\|_1 \leq \epsilon. \tag{2.10}$$

Zum Schluß bestimmen wir die Laufzeit des oben beschriebenen Algorithmus. Laut [LLL82] führt der LLL-Algorithmus $O(r^6 \log^3(\alpha'))$ Bitoperationen aus, wobei α' die Länge des längsten Eingabevektors ist. Da wir den LLL-Algorithmus auf die Matrix $\tilde{\mathsf{A}}$ anwenden und es gilt $\|\tilde{\mathbf{a}}\| = O(2^h\alpha)$, ist die Laufzeit des LLL-Algorithmus $O(r^6 \log^3(2^h\alpha))$. Die Matrixmultiplikation $(\mathbf{a'}_1, \ldots, \mathbf{a'}_{2r})(\mathbf{m}_{r+1}, \ldots, \mathbf{m}_{2r})$ und das Invertieren von B' kann in der Zeit $O(r^3 \log^3(\|\mathsf{B}'\|_1)) = O(r^3 \log^3(2^h r^{\frac{3}{2}}\lambda_r(M_r)))$ berechnet werden. Daraus folgt, daß der ganze Algorithmus in der Zeit

$$O(r^6 \log^3(2^h \max\{\alpha, r^{\frac{3}{2}}\lambda_r(M_r)\}))$$

ausgeführt werden kann. □

Den Algorithmus aus dem letzen Satz nennen wir FindDualBasis.

2.5 Grundlagen der Zahlentheorie

In diesem Kapitel beschreiben wir die Grundlagen der Zahlentheorie, sowie grundlegende zahlentheoretische Algorithmen. Weitere Details zu diesem Thema können in [Coh00] oder [Thi95] nachgelesen werden. Algorithmen für quadratische Zahlkörper werden ausführlich in [BV07] behandelt. Im ersten Teil präsentieren wir die Grundlagen beliebiger Zahlkörper. Im zweiten Teil beschränken wir uns auf die quadratische Zahlkörper.

2.5.1 Beliebige Zahlkörper

Algebraische Zahlen und Zahlkörper

Wir beginnen mit der Definition der algebraischen Zahlen.

Satz 2.5.1 (Algebraische Zahlen) *Sei $n \in \mathbb{N}$. Eine Zahl $\alpha \in \mathbb{C}$ heißt* algebraische Zahl*, wenn es ein Polynom $A = \sum_{i=0}^{n} a_i X^i \in \mathbb{Z}[X]$ mit $a_n \neq 0$ existiert, so daß $A(\alpha) = 0$.*

Die Zahl α heißt ganze algebraische Zahl*, wenn $a_n = 1$ ist.*

Ist A irreduzibel, dann ist n der Grad *von α und A das* Minimalpolynom *von α.*

Als nächstes definieren wir die Konjugieren einer algebraischen Zahl.

Satz 2.5.2 (Konjugierten) *Sei α eine algebraische Zahl und A das Minimalpolynom von α vom Grad n. Dann heißen die anderen Nullstellen von A $\alpha^{(2)}, \ldots, \alpha^{(n-1)}$ die Konjugierten von α.*

Algebraische Zahlen sind unter Addition, Subtraktion, Multiplikation und Division (außer der Division durch Null) abgeschlossen und bilden einen Körper. Dieser Körper ist wie folgt definiert:

Satz 2.5.3 (algebraische Zahlkörper) *Ein* algebraischer Zahlkörper *ist ein Unterkörper von $\mathbb{C}$, welcher $\mathbb{Q}$ enthält und als $\mathbb{Q}$-Vektorraum endlich-dimensional ist. Die Dimension des $\mathbb{Q}$-Vektorraums ist der* Grad *(über $\mathbb{Q}$) des Zahlkörpers.*

Im folgenden, wenn wir von Zahlkörpern sprechen, meinen wir immer algebraische Zahlkörper.

Satz 2.5.4 (primitives Element) *Sei $\mathcal{K}$ ein Zahlkörper vom Grad n. Dann existiert ein Element $\rho \in \mathcal{K}$, so daß $(1, \rho, \rho^2, \ldots, \rho^{n-1})$ eine $\mathbb{Q}$-Basis von $\mathcal{K}$ ist.* □

Für den Körper $\mathcal{K}$ und das Element ρ aus dem letzten Satz schreiben wir abkürzend

$$\mathcal{K} = \mathbb{Q}(\rho)$$

und nennen ρ ein *primitives Element*.

Satz 2.5.5 (erzeugendes Polynom) *Seien $\mathcal{K}$ ein Zahlkörper, $\rho \in \mathcal{K}$ ein primitives Element und $A \in \mathbb{Q}[X]$ das Minimalpolynom von ρ, dann heißt A ein* erzeugendes Polynom *von $\mathcal{K}$.*

Satz 2.5.6 (Einbettungen) *Seien $\mathcal{K}$ ein Zahlkörper vom Grad n und ρ ein primitives Element von $\mathcal{K}$, d.h. $\mathcal{K} = \mathbb{Q}(\rho)$. Dann existieren genau n* Einbettungen *von $\mathcal{K}$ in $\mathbb{C}$, welche durch die Abbildung*

$$\rho \mapsto \rho_i$$

gegeben sind, wobei ρ_i die Nullstellen des Minimalpolynoms von ρ sind. □

Satz 2.5.7 (Signatur) *Die* Signatur *eines Zahlkörpers $\mathcal{K}$ vom Grad n ist ein Paar (s,t). Dabei ist s die Anzahl der Einbettungen von $\mathcal{K}$, deren Bilder in $\mathbb{R}$ liegen, und $2t$ die Anzahl der Einbettungen, deren Bilder in $\mathbb{C}\backslash\mathbb{R}$ liegen und es gilt $s + 2t = n$.*

Ist A ein erzeugendes Polynom von $\mathcal{K}$, dann ist r die Anzahl der reellen Nullstellen und $2s$ die Anzahl der nicht-reellen Nullstellen von A.

Satz 2.5.8 (Archimedische Bewertungen) *Seien $\mathcal{K}$ ein Körper mit Signatur (s,t), $\alpha \in \mathcal{K}$ und $m = s + t$. Seien $\alpha^{(1)}, \ldots, \alpha^{(m)}$ die Konjugierten von α in der Reihenfolge*

$$\alpha^{(1)}, \ldots \alpha^{(s)} \in \mathbb{R} \text{ und } \alpha^{(s+1)}, \ldots, \alpha^{(s+t)}, \overline{\alpha^{(s+1)}}, \ldots, \overline{\alpha^{(s+t)}} \in \mathbb{C}.$$

Die normalisierten archimedischen Bewertungen *für α sind*

$$|\alpha|_i = \begin{cases} |\alpha^{(i)}|, & \text{falls } 1 \le i \le s \text{ und} \\ |\alpha^{(i)}|^2, & \text{falls } s < i \le m \end{cases}.$$

Darstellung

Es gibt mehrere Möglichkeiten zur Darstellung eines Zahlkörpers (siehe z.B. [Coh00], [Len92]). Sie können in Polynomzeit in einander überführt werden. In dieser Arbeit betrachten wir nur die Standard-Darstellung.

Sei $\mathcal{K}$ ein Zahlkörper vom Grad n und ρ ein primitives Element von $\mathcal{K}$, dessen Minimalpolynom $p(X) = \sum_{i=0}^{n} p_i X^i \in \mathbb{Z}[X]$ monisch ist. Aus dem letzten Abschnitt wissen wir, daß $(1, \rho, \ldots, \rho^{n-1})$ eine $\mathbb{Q}$-Basis von $\mathcal{K}$ ist. Daraus folgt, daß jedes Element $\alpha \in \mathcal{K}$ durch die Summe

$$\alpha = \frac{1}{d} \sum_{i=1}^{n} a_i \rho^{i-1} \quad \text{mit } d \in \mathbb{N},\ a_1, \ldots, a_n \in \mathbb{Z} \text{ und } \mathrm{ggT}(a_1, \ldots, a_n, d) = 1 \tag{2.11}$$

eindeutig dargestellt werden kann.

Satz 2.5.9 (Standard-Darstellung) *Die Standard-Darstellung einer Zahl α eines Zahlkörpers $\mathbb{Q}(\rho)$ ist das Tupel*

$$(a_1, \ldots, a_n, d),$$

wobei die Zahlen $d, a_1, \ldots, a_n$ so gewählt sind, daß sie (2.11) erfüllen.

Nun beschreiben wir die grundlegenden Körperoperationen in $\mathcal{K}$. Die Addition und Subtraktion sind einfache Vektorraum-Operationen. Die Multiplikation in $\mathcal{K}$ führen wir in zwei Schritten durch. Zuerst berechnen wir iterativ die Standard-Darstellung von Zahlen $\rho^n, \ldots, \rho^{2n-2}$:

$$\rho^{n+k} = \sum_{i=1}^{n} r_{i,k} \rho^{i-1}.$$

Für $k = 0$ ist die Standard-Darstellung $(-p_0, \ldots, -p_{n-1}, 1)$. Für $0 < k \leq 2n-2$ ist sie $(r_{k,0}, \ldots, r_{k,n-1}, 1)$, wobei

$$r_{k+1,i} = \begin{cases} -t_0 r_{k,n-1}, & \text{falls } i = 0, \\ r_{k,i-1} - t_i r_{k,n-1}, & \text{sonst.} \end{cases}$$

Im zweiten Schritt berechnen wir das Produkt der Zahlen, und zwar seien $\alpha = \frac{1}{d_a} \sum_{i=1}^{n} a_i \rho^i$ und $\beta = \frac{1}{d_b} \sum_{i=1}^{n} b_i \rho^i$, dann ist das Produkt gleich

$$\alpha\beta = \frac{1}{d_a d_b} \sum_{k=0}^{n-1} \left(c_k + \sum_{i=0}^{n-2} r_{k,i} c_{n+i} \right) \rho^k \quad \text{mit } c_k = \sum_{i+j=k} a_i b_j.$$

Die Division in $\mathcal{K}$ ist etwas komplizierter. Um α/β zu berechnen, wird zurerst β^{-1} mit dem erweiterten Euklidischen Algorithmus berechnet. Danach wird die Multiplikation $\alpha\beta^{-1}$, wie oben beschrieben, durchgeführt.

Die beschriebenen Algorithmen führen nun zum folgenden Satz.

Satz 2.5.10 *Es gibt deterministische Polynomzeitalgorithmen, welche bei Eingabe eines Zahlkörpers $\mathcal{K}$ und zwei Elementen $\alpha, \beta \in \mathcal{K}$ die Summe $\alpha + \beta$, die Differenz $\alpha - \beta$, das Produkt $\alpha\beta$ und den Quotienten α/β berechnen.* □

Norm und Höhe

Satz 2.5.11 (Norm) *Seien $\mathcal{K}$ ein Zahlkörper vom Grad n und σ_i die n verschiedenen Einbettungen von $\mathcal{K}$ in $\mathbb{C}$. Für eine Zahl $\alpha \in \mathcal{K}$ definieren wir die* Norm $\mathrm{N}(\alpha)$ *von α als Produkt aller seiner Konjugierten, d.h.*

$$\mathrm{N}(\alpha) = \prod_{i=1}^{n} \sigma_i(\alpha).$$

Die Norm hat folgende Eigenschaften

Satz 2.5.12 *Seien $\mathcal{K}$ ein Zahlkörper und $\alpha, \beta \in \mathcal{K}$. Dann sind folgende Aussagen wahr:*

- $\mathrm{N}(\alpha) \in \mathbb{Q}$. *Ist α eine ganze algebraische Zahl, dann gilt* $\mathrm{N}(\alpha) \in \mathbb{Z}$.
- $\mathrm{N}(\alpha\beta) = \mathrm{N}(\alpha)\,\mathrm{N}(\beta)$.

□

Satz 2.5.13 (Höhe) *Sei $\mathcal{K}$ ein Zahlkörper mit Signatur (s,t). Dann definieren wir die* Höhe *einer Zahl $\alpha \in \mathcal{K}$ wie folgt:*

$$H(\alpha) = \{\, |\alpha|_i \mid 1 \leq i \leq s+t \,\}.$$

Ordnungen

Satz 2.5.14 (Modul) *Eine Untermenge $\mathcal{M}$ eines Zahlkörpers $\mathcal{K}$ vom Grad n heißt ein* Modul *von $\mathcal{K}$, wenn es Elemente $\gamma_1, \ldots, \gamma_n \in \mathcal{M}$ gibt, so daß für jedes $\alpha \in \mathcal{M}$ n eindeutig bestimmte Zahlen $a_1, \ldots, a_n \in \mathbb{Z}$ existieren mit $\alpha = \sum_{i=1}^{n} a_i \gamma_i$.*

Satz 2.5.15 (Ordnung) *Eine Ordnung $\mathcal{O}$ eines Zahlkörper $\mathcal{K}$ ist ein Unterring von $\mathcal{K}$, welcher die Eins enthält und ein Modul von $\mathcal{K}$ ist.*

Ganze algebraische Zahlen

Ähnlich wie die ganzen Zahlen eine Untermenge der rationalen Zahlen bilden, bilden ganze algebraische Zahlen eine Untermenge der algebraischen Zahlen. Die ganzen algebraischen Zahlen eines Zahlkörpers $\mathcal{K}$ bezeichen wir mit $\mathcal{O}_\mathcal{K}$. Für sie gilt der folgende Satz

Satz 2.5.16 *Sei $\mathcal{K}$ ein Zahlkörper vom Grad n. Dann gilt:*

- *$\mathcal{O}_\mathcal{K}$ ist ein Integritätsbereich, d.h. ein nullteilerfreier kommutativer Ring mit Eins und*
- *$\mathcal{O}_\mathcal{K}$ ist ein freier $\mathbb{Z}$-Modull vom Rang n.*

□

Einheiten und Einheitengruppe

Satz 2.5.17 (Einheiten) *Sei $\mathcal{K}$ ein Zahlkörper. Ein Element $\alpha \in \mathcal{O}_\mathcal{K}$ heißt eine* Einheit*, wenn ein $\beta \in \mathcal{O}_\mathcal{K}$ existiert, so daß $\alpha\beta = 1$ gilt.*

Satz 2.5.18 *Sei $\mathcal{K}$ ein Zahlkörper, dann ist $\alpha \in \mathcal{O}_\mathcal{K}$ genau dann eine Einheit, wenn $\mathrm{N}(\alpha) = \pm 1$ ist.* □

Satz 2.5.19 (Einheitengruppe) *Die Einheiten eines Zahlkörpers bilden eine multiplikative abelsche Gruppe $U(\mathcal{K})$, die wir die* Einheitengruppe *nennen. Wenn es klar ist, um welchen Körper es sich handelt, bezeichnen wir die Einheitengruppe mit U.* □

Ein wichtiger Satz über die Einheitengruppe wurde von Dirichlet bewiesen:

Satz 2.5.20 (Dirichlet) *Sei (s,t) die Signatur eines Zahlkörpers $\mathcal{K}$. Sei außerdem d die Anzahl der Einheitswurzel in $\mathcal{K}$. Dann ist die Einheitengruppe $U(\mathcal{K})$ eine endlich erzeugte abelsche Gruppe vom Rang $r = s+t-1$. D.h. es existieren $r+1$ Einheiten $\xi, \nu_1, \ldots, \nu_r$, wobei ξ eine primitive d-te Einheitswurzel in $\mathcal{K}$ ist, so daß jede Einheit $\nu \in U(\mathcal{K})$ eine eindeutige Darstellung*

$$\nu = \xi^{e_0} \prod_{i=1}^{r} \nu_i^{e_i}$$

mit $0 \leq e_0 < d$ und $e_i \in \mathbb{Z}$ für $1 \leq i \leq r$ besitzt. □

Satz 2.5.21 (Fundamentaleinheiten) *Die Einheiten $\nu_1, \ldots, \nu_r$ aus dem letzten Satz heißen* Fundamentaleinheiten. *Die Zahl r heißt der Einheitenrang.*

Satz 2.5.22 (Dirichlet-Abbildung) *Seien $\mathcal{K}$ ein Zahlkörper mit Signatur (s,t) und $r = s+t-1$. Die Abbildung*

$$\mathrm{Log} : \mathcal{K}\backslash\{0\} \to \mathbb{R}^r, \alpha \mapsto \mathrm{Log}\,\alpha = (\ln|\alpha|_1, \ldots, \ln|\alpha|_r)$$

heißt Dirichlet-Abbildung.

Satz 2.5.23 *Seien $\mathcal{O}$ eine Ordnung eines Zahlkörpers $\mathcal{K}$ und $\nu_1, \ldots, \nu_r$ Fundamentaleinheiten. Dann ist das Bild $\mathrm{Log}\,\mathcal{O}^\times$ von $\mathcal{O}^\times$ ein r-dimensionales Gitter in $\mathbb{R}^r$ mit Basis $(\mathrm{Log}\,\nu_1, \ldots, \mathrm{Log}\,\nu_r)$.* □

Satz 2.5.24 (Regulator) *Die Determinante des Gitters aus dem letzten Satz nennen wir* Regulator *von $\mathcal{K}$ und bezeichnen sie mit R.*

Satz 2.5.25 (Diskriminante) *Seien $\mathcal{K}$ ein Zahlkörper vom Grad n und $(\omega_1, \ldots, \omega_n)$ eine Basis von $\mathcal{O}_\mathcal{K}$. Seien außerdem $\sigma_1, \ldots, \sigma_n$ die n Einbettungen von $\mathcal{K}$ in $\mathbb{C}$. Die* Diskriminante *von $\mathcal{K}$ wird mit $d(\mathcal{K})$ bezeichnet und wie folgt definiert*

$$d(\mathcal{K}) = \det(\sigma_i(\omega_j))^2_{1\le i,j\le n}$$

Satz 2.5.26 *Seien $\mathcal{O}$ eine Ordnung eines Zahlkörpers $\mathcal{K}$ vom Grad n mit der Diskriminante Δ, der Signatur (s,t). Sei außerdem $r = s+t-1$. Dann gilt*

$$\lambda_1(\mathrm{Log}\,\mathcal{O}^\times) \ge \max\left\{\left(\frac{2}{n}\right)^{\frac{1}{2}}\left(\frac{1}{2000}\left(\frac{\log\log n}{\log n}\right)^3 - \frac{1}{2880000}\left(\frac{\log\log n}{\log n}\right)^6\right), \frac{16}{17r4^{2+t}}\right\}$$

und

$$\nu(\mathrm{Log}\,U(\mathcal{K})) \le (2r)^r 2^{(n+1)r+2} n^2 \sqrt{|\Delta|}(\log|\Delta|)^{n-1}(\log\log|\Delta|)^{n/2}.$$

Beweis. Siehe [Thi95], Proposition 3.5.29, Seite 25.

Ideale und Klassengruppe

Satz 2.5.27 *Seien $\mathcal{A}$ und $\mathcal{B}$ zwei Teilmengen eines Zahlkörpers $\mathcal{K}$. Dann definieren wir das Produkt von $\mathcal{AB}$ als die Menge*

$$\mathcal{AB} = \{\,\gamma \in \mathcal{K} \mid \gamma = \sum_{i=1}^{k} \alpha_i\beta_i \text{ mit } k \in \mathbb{N},\ a_i \in \mathcal{A} \text{ und } b_i \in \mathcal{B}\,\}$$

Wenn $\mathcal{A} = \{\alpha\}$ eine Menge ist, die nur ein Element enthält, dann schreiben wir für das Produkt $\mathcal{AB}$ einfach $\alpha\mathcal{B}$.

Satz 2.5.28 (Ideale) *Sei $\mathcal{K}$ ein Zahlkörper. Ein (ganzes) $\mathcal{O}_\mathcal{K}$-*Ideal *ist ein Modul $\mathfrak{a} \subseteq \mathcal{O}_\mathcal{K}$, so daß $\mathfrak{a}\mathcal{O}_\mathcal{K} \subseteq \mathfrak{a}$. Ein* gebrochenes $\mathcal{O}_\mathcal{K}$-Ideal *$\mathfrak{a}$ ist ein Modul $\mathfrak{a}$ von $\mathcal{K}$, so daß $n\mathfrak{a}$, $n \in \mathbb{N}$ ein ganzes $\mathcal{O}_\mathcal{K}$-Ideal ist Das minimale n wird der* Nenner *von $\mathfrak{a}$ genannt.*

Sei $(\omega_1, \ldots, \omega_n)$ eine Basis von $\mathcal{O}_\mathcal{K}$. Ein ganzes $\mathcal{O}_\mathcal{K}$-Ideal mit Basis $(\beta_1, \ldots, \beta_n)$ kann als Matrix $\mathsf{A} = (a_{ij}) \in \mathbb{Z}^{n\times n}$ dargestellt werden, wobei $\beta_i = \sum_{j=0}^{n} a_{ij}\omega_j$, $i = 1, \ldots, n$. Die HNF-Form der Matrix A ist die eindeutige Darstellung eines Ideals.

Satz 2.5.29 *Seien $\mathfrak{a}$, $\mathfrak{b}$ $\mathcal{O}_\mathcal{K}$-Ideale und $\alpha \in \mathcal{K}$. Dann existieren Polynomzeitalgorithmen zur Multiplikation von $\mathfrak{a}$ und $\mathfrak{b}$ und zur Multiplikation von $\mathfrak{a}$ und α.* □

Satz 2.5.30 (Teilbarkeit von Idealen, inverse Ideale) *Seien $\mathfrak{a}$ und $\mathfrak{b}$ $\mathcal{O}$-Ideale. Wir sagen* $\mathfrak{a}$ teilt $\mathfrak{b}$*, wenn ein $\mathcal{O}_\mathcal{K}$-Ideal $\mathfrak{c}$ existiert, so daß $\mathfrak{ac} = \mathfrak{b}$ gilt. Außerdem heißt* $\mathfrak{a}$ invertierbar*, falls es ein $\mathcal{O}_\mathcal{K}$-Ideal $\mathfrak{d}$ gibt, so daß $\mathfrak{ad} = \mathcal{O}_\mathcal{K}$.*

Satz 2.5.31 *Invertierbare Ideale bilden eine abelsche Gruppe.* □

Die Gruppe der invertierbaren $\mathcal{O}$-Ideale bezeichnen wir mit $\mathcal{I} = \mathcal{I}_\mathcal{O}$.

Satz 2.5.32 (Äquivalenz von Idealen) *Wir nennen zwei Ideale $\mathfrak{a}$ und $\mathfrak{b}$ von $\mathcal{O}$* äquivalent *und schreiben $\mathfrak{a} \sim \mathfrak{b}$, wenn es ein Element $\alpha \in \mathcal{K}$ gibt, so daß $\mathfrak{a} = \alpha\mathfrak{b}$. Ein solches α nennen wir ein* Relativerzeuger*.*

Satz 2.5.33 (Hauptideale) *Sei $\alpha \in \mathcal{K}$, dann ist $\alpha\mathcal{O}$ ein* Hauptideal *von $\mathcal{O}$. Für $\alpha\mathcal{O}$ schreiben wir abkürzend (α). Die Menge der Hauptideale bezeichnen wir mit $\mathcal{P} = \mathcal{P}_\mathcal{O}$.*

Satz 2.5.34 (Klassengruppe) *Die Faktorgruppe $Cl_\mathcal{O} = \mathcal{I}_\mathcal{O}/\mathcal{P}_\mathcal{O}$ nennen wir die* Klassengruppe *von $\mathcal{O}$, ihre Elemente nennen wir die* Idealklassen *und ihre Kardinalität die* Klassenzahl *von $\mathcal{O}$. Die Klassenzahl bezeichnen wir mit h.*

Minima und reduzierte Ideale

Satz 2.5.35 (Minima, Nachbarn) *Sei $\mathfrak{a}$ ein $\mathcal{O}$-Ideal. Eine Zahl $\mu \in \mathfrak{a}$ heißt* Minimum *von $\mathfrak{a}$, wenn kein $\alpha \in \mathfrak{a}$, $\alpha \neq 0$, existiert, so daß $|\alpha|_i < |\mu|_i$ für alle $1 \leq i \leq m$. Die Menge der Minima eines Ideals $\mathfrak{a}$ bezeichnen wir mit $\mathcal{M}_\mathfrak{a}$. Ein weiteres Minimum $\mu' \in \mathfrak{a}$ heißt* Nachbar *von μ, wenn kein $\alpha \in \mathfrak{a}$, $\alpha \neq 0$, existiert, so daß $|\alpha|_i < \max\{|\mu|_i, |\mu'|_i\}$ für alle $1 \leq i \leq m$.*

Satz 2.5.36 (Reduzierte Ideale) *Ein Ideal $\mathfrak{a}$ heißt* reduziert*, wenn 1 ein Minimum von $\mathfrak{a}$ ist. Die Menge der reduzierten Ideale bezeichnen wir mit $\mathcal{R}$.*

Satz 2.5.37 *Sei $\mathfrak{a}$ ein Ideal. Die reduzierten Ideale, die zu $\mathfrak{a}$ äquivalent sind, sind genau die Ideale $(1/\mu)\mathfrak{a}$, wobei μ ein Minimum von $\mathfrak{a}$ ist.* □

Satz 2.5.38 *Jede Einheit eines Zahlkörpers $\mathcal{K}$ ist ein Minimum von $\mathcal{O}_\mathcal{K}$.* □

Satz 2.5.39 *Seien $\mathcal{K}$ ein Körper, $\mathfrak{a}$ ein $\mathcal{O}_\mathcal{K}$-Ideal und $\alpha \in \mathcal{K}$ eine Einheit. Dann ist μ genau dann ein Minimum von $\mathfrak{a}$, wenn $\alpha\mu$ ein Minimum ist.* □

Satz 2.5.40 *Seien $\mathcal{K}$ ein Zahlkörper vom Grad n, $\mathcal{O}$ eine Ordnung von $\mathcal{K}$ und $\mathfrak{a}$ ein $\mathcal{O}$-Ideal. Dann gibt es einen deterministischen Polynomzeit-Algorithmus, welcher bei Eingabe $\mathcal{O}$ und $\mathfrak{a}$ ein Minimum $\alpha \in \mathfrak{a}$ und ein reduziertes Ideal $\mathfrak{b} = (1/\alpha)\mathfrak{a}$ berechnet.* □

Binäre multiplikative Darstellung (BMD) einer algebraischen Zahl

Satz 2.5.41 (BMD) *Seien $\mathcal{K}$ ein Zahlkörper und $\alpha \in \mathcal{K}$. Dann heißt*

$$M = ((\beta_1, \ldots, \beta_l), (e_1, \ldots, e_l)), \quad \text{mit } l \in \mathbb{N},\ e_i \in \mathbb{Z} \text{ und } \beta_i \in \mathcal{K},\ 1 \leq i \leq l,$$

eine multiplikative Darstellung *von α, wenn*

$$\alpha = \prod_{i=1}^{l} \beta_i^{e_i}.$$

Falls α ein Minimum eines $\mathcal{O}$-Ideals von $\mathcal{K}$ und $e_i = 2^{l-i}$ für alle $1 \leq i \leq l$ ist, dann heißt M die binäre multiplikative Darstellung *(BMD) von α. Für eine BMD von α schreiben wir verkürzend $(\beta_1, \ldots, \beta_l)$.*

Binäre multiplikative Darstellungen sind deshalb interessant, weil damit Minima in reduzierten Idealen kürzer als in der Standarddarstellung repräsentiert werden können. Für solche Minima existieren BMDs mit Einträgen, deren Höhe polynomiell in $\log \Delta$ (bei fixem Körpergrad) ist.

Satz 2.5.42 *Für jedes fixe $\delta > 0$ existiert ein Algorithmus, welcher bei Eingabe einer BMD $(\beta_1, \ldots, \beta_l)$ eines Minimums μ eines reduzierten Ideals ein Vektor $\mathbf{L} = (L_1, \ldots, L_r) \in \mathbb{Q}^r$ berechnet, für das gilt*

$$0 \leq L_i - \ln|\mu|_i < \delta.$$

Die Laufzeit dieses Algorithmus ist $O((l \log \Delta)^{2+\epsilon}(-\log \delta)^{1+\epsilon})$ mit einem kleinen $\epsilon > 0$.

Beweis. Siehe [Thi95], Proposition 6.1.1, Seite 71.

Den Algorithmus aus dem letzten Satz ist in [Thi95] angegeben. Wir nennen ihn ApproxLog.

Ein wichtiges Resultat von Buchmann besagt, dass es möglich ist, alle Minima in der Umgebung eines gegebenen Punktes $\mathbf{v} \in \mathbb{R}^r$ zu enumerieren (siehe [Buc88]).

Satz 2.5.43 *Für $\mathbf{v} \in \mathbb{R}^r$ und $s \in \mathbb{R}_{>0}$ setze*

$$\mathcal{B}(\mathbf{v}, s) = \{\, \mu \in \mathcal{M}_{\mathcal{O}} \mid \|\mathbf{v} - \operatorname{Log} \mu\|_\infty \leq s \,\}.$$

Satz 2.5.44 *Es existiert ein Algorithmus, welcher bei Eingabe eines Vektors $\mathbf{v} \in \mathbb{Q}^r$, einer Ordnung $\mathcal{O}$ und eines invertierbaren $\mathcal{O}$-Ideals $\mathfrak{a}$ die Menge $\mathcal{S}$ aller Minima in $\mathfrak{a}$ berechnet mit*

$$\mathcal{B}(\mathbf{v}, \frac{3 + \ln \Delta}{4}) \subset \mathcal{S} \subset \mathcal{B}(\mathbf{v}, \frac{4 + \ln \Delta}{4}).$$

Dabei werden die Minima in binärer multiplikativen Darstellung $(\beta_1, \ldots, \beta_l)$ mit $l \leq \log\|\mathbf{v}\|_2 + 2$ und $H(\beta_i) \leq (4\Delta)^{2(m+1)}$ ausgegeben. Die Laufzeit des Algorithmus ist polynomiell in $\log \Delta$ und exponentionell im Körpergrad.

Beweis. Siehe [Thi95], Satz 6.2.21, Seite 82.

Der Algorithmus aus dem letzten Satz wird in [Thi95] präsentiert. Wir nennen diesen Algorithmus CollectMinima.

2.5.2 Quadratische Zahlkörper

In diesem Abschnitt beschränken wir uns auf die quadratischen Zahlkörper und beschreiben einige zusätzliche Eigenschaften der Ideale dieser Körper, die wir in den folgenden Kapiteln benötigen werden.

Im weiteren nehmen wir an, daß die Diskriminante Δ entweder negativ ist oder positiv und kein Quadrat ist.

Alle fehlenden Beweise zu den angegebenen Sätzen können in [BV07] nachgelesen werden.

Satz 2.5.45 *Sei Δ die Diskriminante eines quadratischen Körpers. Ist $\Delta < 0$, dann sprechen wir von* imaginär-quadratischen Zahlkörpern. *Ist $\Delta > 0$, dann sprechen wir von* reell-quadratischen *Zahlkörpern.*

Darstellung und Multiplikation von Idealen

Die Ideale in quadratischen Zahlkörpern können als Tripel $(a, b, c) \in \mathbb{Z}^3$ mit $\Delta = b^2 - 4ac$ dargestellt werden. Wenn es klar ist, um welche Diskriminante Δ es sich handelt, können die Ideale als Tupel $(a, b) \in \mathbb{Z}^2$ dargestellt werden.

Die Multiplikation von Idealen ist wie folgt definiert:

Satz 2.5.46 *Seien (a_1, b_1) und (a_2, b_2) zwei Ideale der Diskriminante Δ. Das Produkt (a, b) der Ideale (a_1, b_1) und (a_2, b_2) wird wie folgt bestimmt: Berechne*

$$m = \mathrm{ggT}(a_2, a_1, \frac{b_1 + b_2}{2})$$

und setze

$$a = \frac{a_1 a_2}{m^2}$$

Berechne außerdem $j, k, l \in \mathbb{Z}$, so daß

$$j a_2 + k a_1 + l \frac{b_1 + b_2}{2} = m$$

gilt und setzte

$$b \equiv \frac{j a_2 b_1 + k a_1 b_2 + l(b_1 b_2 + \Delta)/2}{m} \mod 2a.$$

Reduktion von Idealen und Eigenschaften reduzierter Ideale

Satz 2.5.47 (Normales Ideal) *Ein Ideal (a, b, c) der Diskriminante Δ heißt* normal, *wenn gilt*

- $-a < b \leq a$, *falls* $\Delta < 0$ *oder*
- $-|a| < b \leq |a|$, *falls* $\Delta > 0$ *und* $|a| \geq \sqrt{\Delta}$ *oder*
- $\sqrt{\Delta} - 2|a| < b < \sqrt{\Delta}$, *falls* $\Delta > 0$ *und* $|a| < \sqrt{\Delta}$.

Satz 2.5.48 (Reduziertes Ideal) *Ein Ideal (a, b, c) der Diskriminante Δ heißt reduziert, wenn gilt*

- *($\Delta < 0$) (a, b, c) ist normal und $a \leq c$ und $b \geq 0$, falls $a = c$ oder*
- *($\Delta > 0$) $|\sqrt{\Delta} - 2|a|| < b < \sqrt{\Delta}$.*

Die Menge aller reduzierten Ideale der Diskriminante Δ bezeichnen wir mit $\mathcal{R}_\Delta$.

An dieser Stelle ist ein wesentlicher Unterschied zwischen den Äquivalenzklassen der Ideale imaginär- bzw. reell-quadratischer Zahlkörper zu erwähnen. Während im imaginär-quadratischen Fall jede Äquivalenzklasse genau ein reduziertes Ideal enthält, welches als eindeutiger Repräsentant der Äquivalenzklasse verwendet werden kann, enthält eine Äquivalenzklasse eines reell-quadratischen Zahlkörpers im allgemeinen mehrere ($O(\sqrt{\Delta})$) reduzierte Ideale.

Es gilt also der folgende Satz.

Satz 2.5.49 *Jede Äquivalenzklasse eines imaginär-quadratischen Zahlkörpers enthält genau ein reduziertes Ideal.*

Beweis. Siehe [BV07], Theorem 5.5.7, Seite 96.

Als nächstes beschreiben wir die Reduktion von Idealen.

Satz 2.5.50 (Reduktionsoperator) *Sei Δ eine Diskriminante und $\mathcal{O}$ die Ordnung eines quadratischen Zahlkörpers. Der* Reduktionsoperator *ist die Abbildung*

$$\begin{aligned} \rho : \mathcal{I}_\mathcal{O} &\longrightarrow \mathcal{I}_\mathcal{O} \\ (a, b) &\longmapsto (c, b') \end{aligned}$$

mit $c = (b^2 - \Delta)/(4a)$ und $b' = -b + 2sc$ mit

$$s = \begin{cases} \left\lfloor \frac{c+b}{2c} \right\rfloor, & \text{falls } \Delta < 0, \\ \operatorname{sign}(c) \left\lfloor \frac{b+|c|}{2|c|} \right\rfloor, & \text{falls } \Delta > 0 \text{ und } |c| > \sqrt{\Delta}, \\ \operatorname{sign}(c) \left\lfloor \frac{b+\sqrt{\Delta}}{2|c|} \right\rfloor, & \text{falls } \Delta > 0 \text{ und } |c| < \sqrt{\Delta}. \end{cases} \tag{2.12}$$

Ein nicht reduziertes Ideal $\mathfrak{a}$ kann durch mehrfache Anwendung des Reduktionsoperators ρ reduziert werden.

Satz 2.5.51 (Reduktionssequenz) *Sei $\mathfrak{a}$ ein nicht reduziertes $\mathcal{O}_\Delta$-Ideal. Die Folge $\mathfrak{a}, \rho(\mathfrak{a}), \rho(\rho(\mathfrak{a})), \ldots$ heißt* Reduktionssequenz *von $\mathfrak{a}$.*

Satz 2.5.52 *Sei (a, b) ein nicht reduziertes $\mathcal{O}_\Delta$-Ideal. Die Anzahl der Reduktionsschritte ist höchstens*

- $\log_2(a/\sqrt{|\Delta|}) + 2$, *falls $\Delta < 0$ und*
- $\frac{1}{2}\log_2(|a|/\sqrt{\Delta}) + 2$, *falls $\Delta > 0$.*

Beweis. Siehe [BV07] Theorem 5.5.4, Seite 92 und 6.5.3, Seite 114.

Satz 2.5.53 *Seien Δ eine Diskriminante eines quadratischen Zahlkörpers und (a, b) ein nicht reduziertes $\mathcal{O}_\Delta$-Ideal mit $a \leq \sqrt{|\Delta|}$, dann ist $\rho(a, b)$ reduziert.*

Beweis. Siehe [BV07]. Lemma 5.5.3, Seite 92 und 6.5.2, Seite 113.

Satz 2.5.54 *Seien Δ eine Diskriminante und $\mathfrak{a} = (a, b, c)$ ein $\mathcal{O}_\Delta$-Ideal eines quadratischen Zahlkörpers. Seien außerdem $\mathfrak{a}_0 = \mathfrak{a}$ und $\mathfrak{a}_i = \rho(\mathfrak{a}_{i-1})$ für $i = 1, \ldots, n$, wobei $\mathfrak{a}_n$ das erste reduzierte Ideal in der Reduktionssequenz ist, und $s_1, \ldots, s_n$ die dazu gehörenden Normalisierungsfaktoren (siehe Definition 2.5.50). Setze $p_0 = 0$, $p_1 = 1$ und $p_{i+1} = s_i p_i - p_{i-1}$ für $i = 1, \ldots, n$. Dann gilt*

- *$p_i \leq 2K/\sqrt{|\Delta|}$ für alle $i = 1, \ldots n + 1$, falls $\Delta < 0$ oder*
- *$p_i \leq 2K/\sqrt{\Delta}$ für alle $i = 1, \ldots n$ und $p_{n+1} \leq K + K/\sqrt{\Delta}$, falls $\Delta > 0$*

wobei $K = \max\{a, b, c\}$ gilt.

Beweis. Siehe [BV07] Theorem 5.6.1, Seite 92 und Theorem 6.6.1, Seite 115.

Satz 2.5.55 *Sei (a, b, c) ein normales Ideal der Diskriminante Δ mit $a > \sqrt{|\Delta|}$. Dann gilt $c \leq a/2$.*

Beweis. Siehe [BV07], Lemma 5.5.2, Seite 91

Satz 2.5.56 *Seien Δ eine Diskriminante und (a, b, c) ein reduziertes $\mathcal{O}_\Delta$-Ideal. Dann gilt $a \leq \sqrt{|\Delta|/3}$, falls $\Delta < 0$, und $|a| + |c| \leq \sqrt{\Delta}$, falls $\Delta > 0$.*

Beweis. Siehe [BV07], Lemma 5.4.1, Seite 90 und Lemma 6.2.7, Seite 110.

Satz 2.5.57 *Seien $\Delta < 0$ eine Diskriminante, $\mathfrak{a} = (a, b, c)$ ein normales $\mathcal{O}_\Delta$-Ideal mit $a > \sqrt{|\Delta|/3}$ und $c \geq \sqrt{|\Delta|/3}$ und s wie in (2.12), dann gilt $|s| > 1$.*

Beweis. Seien Δ, $\mathfrak{a}$, a, b und c wie im Satz definiert. Zum Beweis zeigen wir, daß die folgende äquivalente Aussage gilt: Falls $|s| \leq 1$, dann ist $c \leq \sqrt{|\Delta|/3}$.

Falls $s = 0$, dann gilt $\rho(\mathfrak{a}) = (c, -b, c)$. Deshalb ist $4c^2 - b^2 = |\Delta|$. Wegen $|b| \leq c$ gilt nun $c \leq \sqrt{|\Delta|/3}$.

Sei nun $s = 1$. Es gilt $\rho(\mathfrak{a}) = (c, -b + 2c, c - b + a)$. Ist $\rho(\mathfrak{a})$ reduziert, dann folgt aus Satz 2.5.56, daß $c \leq \sqrt{|\Delta|/3}$. Ist $\rho(\mathfrak{a})$ nicht reduziert, dann gilt, wegen $|b| \leq a$, $c = c - b + a$. Wie im Fall $s = 0$ folgt daraus $c \leq \sqrt{|\Delta|/3}$.

Der Fall $s = -1$ läßt sich analog zeigen. □

Satz 2.5.58 *Seien $\Delta < 0$ eine Diskriminante, $\mathfrak{a} = (a, b, c)$ ein nicht reduziertes normales $\mathcal{O}_\Delta$-Ideal mit $a \leq \sqrt{|\Delta|/3}$ und s wie in (2.12). Dann gilt $|s| \leq 1$.*

Beweis. Wir verwenden die gleichen Bezeichnungen und Voraussetzungen wie im Satz. Dann gilt

$$c = \frac{b^2 + |\Delta|}{4a} \geq \frac{|\Delta|}{4a} \geq \frac{3}{4}a.$$

Es folgt $|b| \leq a \leq (4/3)c$

Falls $b \geq 0$ ist, dann ist

$$c = \left\lfloor \frac{c+b}{2c} \right\rfloor \leq \left\lfloor \frac{1}{2} + \frac{4c}{3}\frac{1}{2c} \right\rfloor = 1.$$

Falls $b < 0$ ist, dann ist

$$c = \left\lfloor \frac{c+b}{2c} \right\rfloor \geq \left\lfloor \frac{1}{2} - \frac{|b|}{2c} \right\rfloor \geq \left\lfloor \frac{1}{2} - \frac{4c}{3}\frac{1}{2c} \right\rfloor = -1.$$

□

Satz 2.5.59 *Seien $\Delta > 0$ eine Diskriminante und $\mathfrak{a} = (a, b, c)$ ein normales $\mathcal{O}_\Delta$-Ideal. Es gelte $|a|, |c| \geq \sqrt{\Delta}$, dann gilt auch $|s| > 1$.*

Beweis. Wir verwenden die Bezeichnungen und Annahmen wie im Satz. Dann gilt $|a||c| \geq \Delta$. Daraus folgt $4|a||c| \geq 4\Delta$. Wegen $0 \leq b^2 = \Delta + 4ac$, folgern wir, daß $ac > 0$ und deshalb $b^2 \geq 4ac$. Da $|b| \leq |a|$, ist $|a||b| \geq b^2 \geq 4ac$. Aus der letzten Ungleichung folgt $2 \leq b/(2c)$, woraus die Behauptung des Satzes sofort folgt. □

Satz 2.5.60 *Seien $\Delta > 0$ eine Diskriminante und $\mathfrak{a} = (a, b, c)$ ein normales nicht reduziertes $\mathcal{O}_\Delta$-Ideal. Es gelte $|a| \leq \sqrt{\Delta}$, dann gilt auch $|s| < \sqrt{\Delta} + 1$.*

Beweis. Wir verwenden die gleichen Bezeichnungen und Annahmen wie im Satz. Nach Satz 2.5.53 ist $\rho(\mathfrak{a})$ reduziert, mit Satz 2.5.56 folgt daraus, daß $|c| \leq \sqrt{\Delta}$ gilt. Wir erhalten

$$|s| \leq \left| \left\lfloor \frac{b + \sqrt{\Delta}}{2c} \right\rfloor \right| \leq \frac{|b|}{2} + \frac{\sqrt{\Delta}}{2} + 1 \leq \sqrt{\Delta} + 1.$$

□

Schranken für die Klassenzahl und den Regulator

Die folgenden zwei Sätze beschreiben eine obere Schranke für die Klassenzahl und den Regulator.

Satz 2.5.61 *Sei $\Delta < -4$ eine Diskriminante, dann gilt für die Klassenzahl h_Δ*

$$h_\Delta < \frac{\sqrt{|\Delta|}\ln|\Delta|}{3}.$$

Beweis. Siehe [Sla69].

Satz 2.5.62 *Sei $\Delta \geq 5$ eine Diskriminante, dann gilt für die Klassenzahl h_Δ und Regulator R*

$$h_\Delta R < \frac{\sqrt{\Delta}}{2}(\frac{1}{2}\ln\Delta + 1).$$

Beweis. Siehe [Hua82].

Infrastruktur

Wie bereits erwähnt, enthält eine Äquivalenzklasse eines reell-quadratischen Zahlkörpers im allgemeinen mehrere reduzierte Ideale. Diese Ideale können auf einem Kreis mit Umfang R angeordnet werden, wobei R der Regulator des Zahlkörpers ist. In diesem Abschnitt beschreiben wir diese Anordnung. Weitere Informationen können in [Len82] gefunden werden.

Sei $\Delta > 0$ für den Rest dieses Abschnitts.

Satz 2.5.63 *Sei $\mathfrak{a} = (a, b, c)$ ein $\mathcal{O}_\Delta$-Ideal. Setze*

$$\gamma(\mathfrak{a}) = \frac{b + \sqrt{\Delta}}{2c}.$$

Satz 2.5.64 *Es gilt $\rho(\mathfrak{a}) = (1/\gamma(\mathfrak{a}))\mathfrak{a}$.*

Beweis. Siehe [BV07] Lemma 9.1.9, Seite 179.

Satz 2.5.65 *Ist $\mathfrak{a}$ ein reduziertes $\mathcal{O}_\Delta$-Ideal, so ist auch $\rho(\mathfrak{a})$ ein reduziertes Ideal.* □

Wenn wir für die Reduktion die Formel aus der Definition 2.5.50 verwenden, werden wir festellen, daß das Vorzeichein von a sich jedes mal ändert.

Satz 2.5.66 *Die Menge der reduzierten Ideale mit $a > 0$ bezeichnen wir mit $\mathcal{R}^+$.*

Satz 2.5.67 (Abstand) *Sei $\alpha \in \mathcal{K}$, dann definiere*

$$\operatorname{Log} \alpha = \frac{1}{2} \ln \left| \frac{\sigma\alpha}{\alpha} \right|.$$

Es gilt offensichtlich $\operatorname{Log}(\alpha\beta) = \operatorname{Log} \alpha + \operatorname{Log} \beta$.

Satz 2.5.68 (Abstand zwischen Idealen) *Sei $\mathcal{K}$ ein reel-quadratischer Zahlkörper der Diskriminante Δ. Seien außerdem $\mathfrak{a}$ und $\mathfrak{b}$ zwei $\mathcal{O}_\Delta$-Ideale und es gelte $\mathfrak{a} = \gamma\mathfrak{b}$ mit $\gamma \in \mathcal{K}$, dann ist der Abstand zwischen $\mathfrak{a}$ und $\mathfrak{b}$ wie folgt definiert:* $\operatorname{dist}(\mathfrak{a}, \mathfrak{b}) = \operatorname{Log} \gamma \mod R$.

Mit der dist-Abbildung lassen sich reduzierte Ideale einer Äquivalenzklasse auf einem Kreis mit Umfang R anordnen. In diesem Zusammenhang ist die folgende Definition nützlich.

Satz 2.5.69 (rechter Nachbar, linker Nachbar) *Sei $\mathfrak{a}$ ein reduziertes $\mathcal{O}_\Delta$-Ideal, dann ist $\rho(\mathfrak{a})$ der* rechte Nachbar *von $\mathfrak{a}$ und $\mathfrak{a}$ der* linke Nachbar *von $\rho(\mathfrak{a})$.*

Für Hauptideale definieren wir die folgende kürzere Schreibweisen.

Satz 2.5.70 *Seien $x \in \mathbb{R}$ und $\mathfrak{a} = \alpha\mathcal{O}_\Delta$, $\alpha \in \mathcal{K}$, ein Hauptideal, dann definiere*

$$\operatorname{Log} \mathfrak{a} = \operatorname{Log} \alpha \mod R$$

und

$$\operatorname{dist}(\mathfrak{a}, x) = x - \operatorname{Log} \alpha \mod R$$

Satz 2.5.71 *Seien $x \in \mathbb{R}_+$ und $d_1 = \operatorname{dist}(\mathcal{O}_\Delta, \rho(\mathcal{O}_\Delta))$, $d_2 = \operatorname{dist}(\rho(\mathcal{O}_\Delta), \rho(\rho(\mathcal{O}_\Delta)))$, ..., dann ist $\mathfrak{a}_-(x)$,* das Ideal links von *x, wie folgt definiert: $\mathfrak{a}_-(x) = \rho^n(\mathcal{O}_\Delta)$, wobei $n \in \mathbb{N}$ so gewählt ist, daß $\sum_{i=1}^{n} d_i \leq x$ und $\sum_{i=1}^{n+1} d_i > x$ gilt.*

Das Ideal $\mathfrak{a}_-(x)$ kann nicht immer in Polynomzeit berechnet werden, da zu seiner Bestimmung Logarithmen ausgerechnet werden müssen. Aus diesem Grund verwenden wir in den folgenden Kapiteln eine approximative Version des linken Ideals. Dazu wählen einen Algorithmus Ln, welcher die Funktion Log approximativ berechnet, und verwenden diesen Algorithmus immer anstatt der exakten Funktion Log.

Für den Abstand zwischen zwei Idealen gelten die folgenden Sätze.

Satz 2.5.72 *Sei $\mathfrak{a}$ ein reduziertes Ideal, dann gilt*

- $1/\sqrt{\Delta} < -\operatorname{Log}\gamma(\mathfrak{a}) \leq \frac{1}{2}\ln\Delta$
- $\ln 2 \leq -\operatorname{Log}\gamma(\mathfrak{a})\gamma(\rho(\mathfrak{a}))$

Beweis. Siehe [BV07] Lemma 10.1.6, Seite 220.

Satz 2.5.73 *Sei $\mathfrak{a} \in \mathcal{I}_\Delta$ ein invertierbares $\mathcal{O}$-Ideal. Außerdem sei $\mathfrak{b}$ das erste reduzierte Ideal in der Reduktionssequenz von $\mathfrak{a}$ und γ ein Erzeuger von $\mathfrak{b}$ relativ zu $\mathfrak{a}$ (siehe Definition 2.5.32), dann gilt*

$$|\operatorname{Log}\gamma| \leq \frac{1}{2}\ln\Delta.$$

Außerdem gilt für $\gamma_1 = \gamma\gamma(\mathfrak{b})^{-1}\gamma(\rho(\mathfrak{b}))^{-1}$ und $\gamma_2 = \gamma\gamma'(\mathfrak{b})^{-1}\gamma'(\rho^{-1}(\mathfrak{b}))^{-1}$

$$\begin{aligned} \rho^2(\mathfrak{b}) &= \gamma_1\mathfrak{a}, \quad \textit{mit } \operatorname{Log}\gamma_1 \geq 0 \textit{ und} \\ \rho^{-2}(\mathfrak{b}) &= \gamma_2\mathfrak{a}, \quad \textit{mit } \operatorname{Log}\gamma_2 \leq 0 \textit{ und} \end{aligned}$$

Beweis. Siehe [Len82].

Kapitel 3

Arithmetische Operationen mit Quantencomputern

In diesem Kapitel werden die grundlegenden arithmetischen Operationen beschrieben, die als Basis für Algorithmen in späteren Kapiteln verwendet werden. Eine ausführliche Einführung in das Thema "Quantencomputer" sowie weitere tiefer gehende Informationen hierzu können dem Buch von Nielsen und Chuang [NC00] entnommen werden.

Die Algorithmen in diesem Kapitel werden mit dem Ziel entwickelt, möglichst wenige Qubits für ihre Ausführung zu gebrauchen. Eine Schwierigkeit, die bei der Entwicklung von Quantenalgorithmen auftritt, beruht darauf, daß alle Operationen im Quantencomputermodell reversibel sein müssen, so daß gewöhnliche Algorithmen, die für klassische Computer entwickelt wurden, nicht eins zu eins übernommen werden können. Dieses Problem kann gelöst werden, indem man zusätzliche Qubits verwendet, in denen Daten gespeichert werden, die die Reversibilität sicherstellen (siehe [Ben77], [NC00]). Die Anzahl solcher Daten kann jedoch sehr schnell wachsen. Deshalb muß in jedem Algorithmus ein Weg gefunden werden, wie man diese Daten reduziert. Dies kann durch die folgenden Techniken erreicht werden:

- durch die Wahl geeigneter Algorithmen (z.B. fallen in der Standard-GGT-Methode weniger temporäre Qubits an als in der binären GGT-Methode),
- dadurch, daß Algorithmen so in Teilalgorithmen zerlegt werden, daß nach der Ausführung eines jeden Teils, alle oder einige dabei anfallende temporäre Qubits durch geeignete Berechnungen gelöscht werden können,
- durch das Auslagern einiger Berechnungen aus dem Quantencomputer in den klassischen Computer.

Als grundlegende Quantenoperationen werden in dieser Arbeit das H-, cNOT-, cR_k- und Toffoly-Gatter verwendet. Das H-Gatter ermöglicht die Herstellung einer Superposition, einer der wichtigsten Eigenschaft, die die Quantencomputer von klassischen Computern unterscheidet. cNOT- und Toffoly-Gatter werden zur Nachbildung klassischer Logik-Operationen (nicht, und, oder etc.) verwendet. Wie in [NC00] gezeigt wird, läßt sich jeder

klassische Algorithmus allein mit dem TOFFOLY-Gatter in einem Quantencomputer implementieren. Schließlich wird das cR_k-Gatter für die Berechnung der Fouriertransformation verwendet.

Die wichtigsten Algorithmen bzw. Modifikationen bekannter Algorithmen, die in dieser Arbeit neu entwickelt und analysiert wurden, sind: cMULT-B, XGCD, cLN_{ADD} und cLN_{ADD}.

Das Kapitel ist wie folgt aufgebaut: Im ersten Abschnitt werden die Begriffe "Qubits" und "Quantengatter" definiert. Im zweiten Abschnitt wird beschrieben, wie die Zahlen in klassischen und Quantenregistern dargestellt werden. Im dritten Abschnitt werden alle elementare Quantengatter beschrieben, die für Algorithmen in dieser Arbeit notwendig sind. Im nächsten Abschnitt wird die Quantenfouriertransformation beschrieben, ein zentraler Algorithmus, welcher ermöglicht, daß das Faktorisierungsproblem sowie das DL-Problem in endlichen abelschen Gruppen auf einem Quantencomputer in Polynomzeit gelöst werden können. Im Abschnitt fünf werden Algorithmen zum Kopieren und Vertauschen von Quantenregistern präsentiert. In den Abschnitten fünf bis dreizehn werden grundlegende arithmetische Operationen (Addition, Subtraktion, Multiplikation, Division, Vergleich) beschrieben. Im Abschnitt 14 wird ein Algorithmus zur Berechnung des größten gemeinsamen Teilers zweier ganzen Zahlen vorgestellt. Im Abschnitt 15 wird ein Algorithmus zur Berechnung des natürlichen Logarithmus entwickelt. Im nächsten Abschnitt wird ein Framework zur Berechnung des Funktionengitters einer Funktion angegeben. Schließlich wird im letzten Abschnitt das Konzept der semi-klassischen Quantenfouriertransformation vorgestellt, welcher die Anzahl der Qubits, die für die Berechnungen gebraucht werden, stark reduziert.

3.1 Quantenbits und Quantengatter

In diesem Abschnitt stellen wir zuerst eine mathematische Definition eines Quantenbits (kurz Qubits) vor. Für die Darstellung der Qubits verwenden wir die in Physik übliche Bra-Ket-Notation $|\ldots\rangle$.

Satz 3.1.1 (Quantenbit, Qubit) *Sei $(|0\rangle, |1\rangle)$ eine Orthonormalbasis des Hilbertraums $\mathbb{C}^2$. Dann ist ein Quantenbit*

$$\phi = a|0\rangle + b|1\rangle, \quad a, b \in \mathbb{C}$$

ein Einheitsvektor in $\mathbb{C}^2$, d.h. es gilt $|a| + |b| = 1$.

Das Qubit $|0\rangle$ stellt die Zahl Null dar. Das Qubit $|1\rangle$ stellt die Zahl Eins dar. Wie man aus der Definition 3.1.1 sieht, kann ein Qubit auch eine Superposition der beiden Zustände, $|0\rangle$ und $|1\rangle$, darstellen.

Satz 3.1.2 *Ein System aus mehreren Qubits*

$$\phi = \phi_0 \otimes \cdots \otimes \phi_n$$

ist ein Tensorprodukt von n Qubits $\phi_0, \ldots, \phi_n$. Es ist ein Einheitsvektor im Hilbertraum $\mathbb{C}^{(2^n)}$.

Sei $i_0 i_1 \dots i_{n-1}$ die Binärdarstellung einer natürlichen Zahl i, dann stellt das n-Qubit-System $|i\rangle = |i_0\rangle \otimes \cdots \otimes |i_{n-1}\rangle$ die Zahl i dar. Eine Superposition mehrerer Werte läßt sich folgendermaßen darstellen:

$$\sum_{i=0}^{2^n-1} a_i |i\rangle, \quad \text{mit } a_i \in \mathbb{C} \text{ und } \sum_{i=0}^{2^n-1} |a_i| = 1. \tag{3.1}$$

Berechnungen auf dieser Superposition werden mit quantenmechanischen Operationen durchgeführt. Es sind nur unitäre Operationen erlaubt. Diese Operationen können mit unitären Matrizen beschreiben werden. Das bedeutet, daß alle Berechnungen auf Quantencomputern reversivel sind. Für reversible Operationen verwenden wir die folgende Notatation:

Satz 3.1.3 *Falls* X *ein Quantenalgorithmus ist, welcher durch eine unitäre Matrix* U *beschrieben wird, dann beschreibt die Matrix* $\mathsf{M}^\dagger = (\mathsf{M}^*)^T$ *den Algorithmus* $\mathrm{X}^\dagger$.

Die Matrizen der elementaren Operationen präsentieren wir im dritten Abschnitt.

Eine *Messung* der Superposition (3.1) liefert einen klassischen Wert. Es wird dabei ein zufälliges i, $0 \le i < n$, mit Wahrscheinlichkeit $|a_i|^2$ gemessen.

3.2 Darstellung der Zahlen und Notationen

Um Zahlen in Quantenregistern darzustellen, verwenden wir die Zweier-Kompliment-Darstellung. In einem n-Bit-Quantenregister wird eine positive ganze Zahl

$$x = \sum_{k=0}^{n-1} x_k 2^k, \quad x_0, \dots, x_{n-1} \in \{0, 1\},$$

als Tupel $(x_{n-1}, \dots, x_0)$ gespeichert. Eine negative ganze Zahl y wir als Tupel $(y_{n-1}, \dots, y_0)$ gespeichert, wobei gilt

$$2^n + y = \sum_{k=0}^{n-1} y_k 2^k, \quad y_0, \dots, y_{n-1} \in \{0, 1\}$$

.

Für die rationale Zahlen verwenden wir die Festpunkt-Darstellung: In einem $(n+m)$-Bit-Quantenregister wird eine Zahl

$$x = \sum_{k=-m}^{n-1} x_k 2^k, \quad x_{-m}, \dots, x_{n-1} \in \{0, 1\}$$

als Tupel $(x_{n-1}, \dots, x_{-m})$ gespeichert. Für negative Zahlen verwenden wir auch hier die Zweier-Komplement-Darstellung.

In den folgenden Algorithmen verwenden wir sowohl Daten, die in klassischen Registern gespeichert werden, als auch Daten, die in Quantenregister gespeichert werden. Um die beiden Fälle zu unterscheiden, benutzen wir die folgende Notation: Wir schreiben

x, falls die Zahl in einem klassischen Register gespeichert werden kann

und

$|x\rangle$, falls die Zahl in einem Quantenregister gespeichert werden muß.

3.3 Elementare Gatter

Wir beginnen mit der Beschreibung der elementaren Gatter (siehe [NC00] für mehr Informationen). Mit Hilfe dieser Gatter können alle Quantenalgorithmen realisiert werden, die wir in den folgenden Kapiteln vorstellen. Die Anzahl der verwendeten Gatter zeigt uns dabei die Zeitkomplexität eines solchen Algorithmus an.

Wir beginnen mit dem Hadamard-Gatter. Es wird auf ein Qubit angewendet. Es wird hauptsächlich dazu verwendet, um aus einem Initialzustand $|0\rangle$ eine Superposition über alle möglichen Werte zu erzeugen. Die Matrix H beschreibt die Wirkung des Gatters. Die Tabelle zeigt diese Wirkung auf die Basiszustände.

$$\mathrm{H} = \frac{\sqrt{2}}{2}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

Eingabe	Ausgabe
$\|0\rangle$	$\sqrt{2}/2(\|0\rangle + \|1\rangle)$
$\|1\rangle$	$\sqrt{2}/2(\|0\rangle - \|1\rangle)$

Das nächste Ein-Qubit-Gatter ist das R_k-Gatter.

$$\mathrm{R}_k = \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{pmatrix}$$

Eingabe	Ausgabe
$\|0\rangle$	$\|0\rangle$
$\|1\rangle$	$e^{2\pi i/2^k}\|1\rangle$

Dieses Gatter werden wir in der gesteuerten Version einsetzen: Das cR_k-Gatter ist wie folgt definiert:

$$\mathrm{cR}_k = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{\pi i/2^k} \end{pmatrix}$$

Eingabe	Ausgabe
$\|00\rangle$	$\|00\rangle$
$\|01\rangle$	$\|01\rangle$
$\|10\rangle$	$\|10\rangle$
$\|11\rangle$	$e^{2\pi i/2^k}\|11\rangle$

Das cR_k-Gatter operiert auf zwei Qubits. Das erste Qubit ist das Steuerungsqubit, das zweite Qubit ist das Zielqubit. Die Wirkung dieses Gatters ist die folgende: Falls das Steuerungsqubit gleich Eins ist, wird die Operation R_k auf das Zielqubit angewendet. Ist das Steuerungsqubit gleich Null, so wird das Zielqubit nicht verändert. Das cR_k-Gatter wird bei der Quantenfouriertransformation und bei der Quantenaddition verwendet.

Ein anderes Gatter, das auf zwei Qubits operiert, ist das cNOT-Gatter. Die dazu gehörende Matrix sieht wie folgt aus:

$$\mathrm{cNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Eingabe	Ausgabe
$\|00\rangle$	$\|00\rangle$
$\|01\rangle$	$\|01\rangle$
$\|10\rangle$	$\|11\rangle$
$\|11\rangle$	$\|10\rangle$

Das letzte Gatter, das wir hier vorstellen, heißt Toffoly-Gatter. Es operiert auf drei Qubits und wird durch die folgende Matrix beschrieben:

$$\textsc{Toffoly} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Eingabe	Ausgabe
$\|000\rangle$	$\|000\rangle$
$\|001\rangle$	$\|001\rangle$
$\|010\rangle$	$\|010\rangle$
$\|011\rangle$	$\|011\rangle$
$\|100\rangle$	$\|100\rangle$
$\|101\rangle$	$\|101\rangle$
$\|110\rangle$	$\|111\rangle$
$\|111\rangle$	$\|110\rangle$

Mit dem Toffoly-Gatter können alle klassischen Operationen realisiert werden (siehe [NC00], Abschnitt 1.4, Seite 29).

3.4 Quantenfouriertransformation

In diesem Abschnitt definieren wir die Quantenfouriertransformation und geben einen effizienten Quantenalgorithmus zu ihrer Berechnung an.

Satz 3.4.1 (QFT) *Sei* $(|0\rangle, \ldots, |2^n - 1\rangle)$ *die kanonische Basis eines Hilbert-Raums* $\mathbb{C}^{(2^n)}$. *Seien außerdem* $a_0, \ldots, a_{2^n-1} \in \mathbb{C}$ *mit* $\sum_{x=0}^{2^n-1} |a_x|^2 = 1$. *Dann ist die Quantenfouriertransformation (*QFT*) wie folgt definiert:*

$$\begin{aligned} \mathrm{QFT}: \quad \mathbb{C}^{(2^n)} \quad &\longrightarrow \quad \mathbb{C}^{(2^n)} \\ \sum_{x=0}^{2^n-1} a_x |x\rangle &\longmapsto \frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} \sum_{x=0}^{2^n-1} a_x e^{2\pi i x y / 2^n} |y\rangle. \end{aligned}$$

Lemma 3.4.2 *Es gilt*

$$\sum_{y=0}^{2^n-1} e^{2\pi i x y / 2^n} |y\rangle = (|0\rangle + e^{2\pi i 0.x_n}|1\rangle)(|0\rangle + e^{2\pi i 0,x_{n-1}x_n}|1\rangle) \cdots (|0\rangle + e^{2\pi i 0,x_1 x_2 \cdots x_n}|1\rangle)$$

mit $0, x_1 = \frac{x_1}{2}, \ \ldots, \ 0, x_1 x_2 \cdots x_n = \sum_{i=1}^{n} 2^{-i} x_i$.

Beweis. Siehe [NC00], Abschnitt 5.1, Seite 218.

Basierend auf diesem Lemma können wir die Quantenfouriertransformation mit Hilfe von H- und $\textsc{cR}_k$-Gattern wie folgt berechnen:

Algorithm 1 QFT

Input: $|x\rangle$, mit $x = \sum_{i=1}^{n} 2^{i-1} x_i$.
Output: $|y\rangle$, mit $|\sum_{i=1}^{n} 2^{n-i+1} y_i\rangle = \mathrm{QFT}(|x\rangle)$.

for $i = 1, \ldots, n$ **do**
  $|x_i\rangle \xrightarrow{\mathrm{H}} |x_i'\rangle$
  for $j = i+1, \ldots, n$ **do**
    $|x_j\rangle, |x_i'\rangle \xrightarrow{\mathrm{cR}_{j-i+1}} |x_j\rangle, |y_i\rangle$

Die Zeit- bzw. Speicher-Komplexität des Algorithmus QFT ist in der folgenden Tabelle zusammengefaßt.

	Anzahl der Qubits		Anzahl der Gatter	
Funktion	Ein-/Ausgabe	Intern	H	cR_k
QFT	n	0	n	$n(n-1)/2$

Tabelle 3.1: Qubit- und Gatter-Komplexität von QFT

Wie man sieht, ist dies ein Polynomzeit-Quantenalgorithmus. Der beste klassische Algorithmus braucht dagegen $\Theta(n2^n)$ Gatter. Dieser exponentielle Unterschied ist der Grund dafür, daß das Faktorisierungs- und das DL-Problem, welche im klassischen Computermodell als schwer gelten, im Quantencomputer-Modell in Polynomzeit gelöst werden können.

3.5 Funktionen cCopy und cSwap

In diesem Abschnitt stellen wir zwei Hilfsalgorithmen zum Kopieren und Vertauschen von Werten in zwei Quantenregistern vor (siehe [NC00]).

Algorithm 2 cCopy

Input: $|\mathsf{ctrl}\rangle$, $|x\rangle$.
Output: $|\mathsf{ctrl}\rangle$, $|x\rangle$, $|x'\rangle$, wobei $x' = 0$, falls $\mathsf{ctrl} = 0$ ist, und $x' = x$, falls $\mathsf{ctrl} = 1$ gilt.

for $i = 1, \ldots, n$ **do**
  $|\mathsf{ctrl}\rangle, |x_i\rangle, |0\rangle \xrightarrow{\textsc{Toffoly}} |\mathsf{ctrl}\rangle, |x_i\rangle, |x_i'\rangle$

Algorithm 3 cSwap

Input: $|\mathsf{ctrl}\rangle$, $|x\rangle$, $|x'\rangle$.
Output: $|\mathsf{ctrl}\rangle$, $|x'\rangle$, $|y'\rangle$, wobei $x' = x$ und $y' = y$, falls $\mathsf{ctrl} = 0$ ist, und $x' = y$ und $y' = x$, falls $\mathsf{ctrl} = 1$ gilt.

for $i = 1, \ldots, n$ **do**

$$\begin{aligned} |y_i\rangle, |x_i\rangle &\xrightarrow{\text{cNOT}} |y_i\rangle, |a\rangle \\ |\mathsf{ctrl}\rangle, |a\rangle, |y_i\rangle &\xrightarrow{\text{Toffoly}} |\mathsf{ctrl}\rangle, |a\rangle, |b\rangle \\ |b\rangle, |a\rangle &\xrightarrow{\text{cNOT}} |y_i'\rangle, |x_i'\rangle \end{aligned}$$

Die Anzahl der verwendeten Qubits und Gatter ist die folgende:

	Anzahl der Qubits		Anzahl der Gatter	
Funktion	Ein-/Ausgabe	Intern	cNOT	Toffoly
cCopy	$2n+1$	0	0	n
cSwap	$2n+1$	0	$2n$	n

Tabelle 3.2: Qubit- und Gatter-Komplexität von cCopy und cSwap

3.6 Addition

In diesem Abschnitt präsentieren wir vier Algorithmen zur Addition zweier n-Bit Zahlen. Wir unterscheiden dabei zwischen der normalen und der gesteuerte Addition und außerdem zwischen der klassischen Addition siehe ([VAE96]) und der Quantenaddition (siehe [Dra00])

Die normale Addition führt die folgende Transformation aus:

$$|a\rangle, |b\rangle, |\mathsf{carry}_{in}\rangle \xrightarrow{\text{Add}} |a\rangle, |b'\rangle, |\mathsf{carry}_{out}\rangle,$$

wobei gilt $a, b, b' \in \mathbb{N}$, $0 \leq a, b, b' < 2^n$, $b' = a + b \mod 2^n$ und

$$\mathsf{carry}_{out} = \begin{cases} \mathsf{carry}_{in}, & \text{falls } a + b < 2^n \\ \mathsf{carry}_{in} + 1 \mod 2, & \text{sonst} \end{cases}$$

Nützlich für die unten vorgestellten Algorithmen ist auch die gesteuerte Version des Additionsalgorithmus:

$$|\mathsf{ctrl}\rangle, |a\rangle, |b\rangle, |\mathsf{carry}_{in}\rangle \xrightarrow{\text{cAdd}} |\mathsf{ctrl}\rangle, |a\rangle, |b'\rangle, |\mathsf{carry}_{out}\rangle, \tag{3.2}$$

wobei gilt $a, b, b' \in Z$, $0 \leq a, b, b' < 2^n$ und

1. $b' = a + b \mod 2^n$ und $\mathsf{carry}_{out} = \mathsf{carry}_{in}$, falls $\mathsf{ctrl} = 1$ und $a + b < 2^n$,
2. $b' = a + b \mod 2^n$ und $\mathsf{carry}_{out} = \mathsf{carry}_{in} + 1 \mod 2$, falls $\mathsf{ctrl} = 1$ und $a + b \geq 2^n$,
3. $a' = a$ und $\mathsf{carry}_{out} = \mathsf{carry}_{in}$, falls $\mathsf{ctrl} = 0$.

3.6.1 Hilfsgatter

Für den Additionsalgorithmus benötigen wir zusätzliche Hilfsgatter, die wir in diesem Abschnitt präsentieren.

Das erste Gatter ist das CARRY-Gatter. Es operiert auf vier Qubits und wird verwendet, um festzustellen, ob bei der Addition von drei Bits ein Übertrag auftritt. Es führt also die folgende Operation aus:

$$|q_1\rangle, |q_2\rangle, |q_3\rangle, |q_4\rangle \overset{\text{CARRY}}{\longrightarrow} |q_1\rangle, |q_2\rangle, |q_3'\rangle, |q_4'\rangle$$

mit $q_3' = q_3 \oplus q_2$ und $q_4' = q_4 \oplus (q_1 \wedge q_2 \vee q_1 \wedge q_3 \vee q_2 \wedge q_3)$.

Es kann mit einem cNOT und zwei TOFFOLY-Gattern realisiert werden.

Algorithm 4 CARRY

Input: $|q_1\rangle, |q_2\rangle, |q_3\rangle, |q_4\rangle$.
Output: $|q_1\rangle, |q_2\rangle, |q_3'\rangle, |q_4'\rangle$.

$$\begin{aligned}
|q_2\rangle, |q_3\rangle, |q_4\rangle &\overset{\text{TOFFOLY}}{\longrightarrow} |q_2\rangle, |q_3'\rangle, |q_4''\rangle \\
|q_2\rangle, |q_3\rangle &\overset{\text{cNOT}}{\longrightarrow} |q_2\rangle, |q_3'\rangle \\
|q_1\rangle, |q_3'\rangle, |q_4''\rangle &\overset{\text{TOFFOLY}}{\longrightarrow} |q_1\rangle, |q_3'\rangle, |q_4'\rangle
\end{aligned}$$

Die gesteuerte Version cCARRY kann mit zwei cNOT-Gattern und zwei TOFFOLY-Gattern implementiert werden. Sie führt die folgende Operation aus:

$$|q_0\rangle, |q_1\rangle, |q_2\rangle, |q_3\rangle, |q_4\rangle \overset{\text{CARRY}}{\longrightarrow} |q_0\rangle, |q_1'\rangle, |q_2'\rangle, |q_3'\rangle, |q_4'\rangle$$

mit $q_1' = q_1 \oplus q_3$, $q_2' = q_2 \oplus q_3$, $q_3' = q_1 \wedge q_2 \vee q_1 \wedge q_3 \vee q_2 \wedge q_3$ und $q_4' = q_4 \oplus (q_0 \wedge (q_1 \wedge q_2 \vee q_1 \wedge q_3 \vee q_2 \wedge q_3))$.

Algorithm 5 cCARRY

Input: $|q_0\rangle, |q_1\rangle, |q_2\rangle, |q_3\rangle, |q_4\rangle$.
Output: $|q_0\rangle, |q_1'\rangle, |q_2'\rangle, |q_3'\rangle, |q_4'\rangle$.

$$\begin{aligned}
|q_3\rangle, |q_2\rangle &\overset{\text{cNOT}}{\longrightarrow} |q_3\rangle, |q_2'\rangle \\
|q_3\rangle, |q_1\rangle &\overset{\text{cNOT}}{\longrightarrow} |q_3\rangle, |q_1'\rangle \\
|q_1'\rangle, |q_2'\rangle, |q_3\rangle &\overset{\text{TOFFOLY}}{\longrightarrow} |q_1'\rangle, |q_2'\rangle, |q_3'\rangle \\
|q_0\rangle, |q_3'\rangle, |q_4\rangle &\overset{\text{TOFFOLY}}{\longrightarrow} |q_0\rangle, |q_3'\rangle, |q_4'\rangle
\end{aligned}$$

Die anderen zwei Gatter, die wir als nächstes vorstellen, berechnen die Summe von drei Bits modulo Zwei. Es sind das SUM-Gatter, welches bei Eingabe von drei Qubits ihre Summe modulo Zwei berechnet, und die gesteuerte Version: das cSUM-Gatter.

Algorithm 6 SUM

Input: $|q_1\rangle, |q_2\rangle, |q_3\rangle$.
Output: $|q_1\rangle, |q_2\rangle, |q_3'\rangle$.

$|q_2\rangle, |q_3\rangle \xrightarrow{\text{cNOT}} |q_2\rangle, |q_3''\rangle$
$|q_1\rangle, |q_3''\rangle \xrightarrow{\text{cNOT}} |q_1\rangle, |q_3'\rangle$

Algorithm 7 CSUM

Input: $|q_0\rangle, |q_1\rangle, |q_2\rangle, |q_3\rangle$.
Output: $|q_0\rangle, |q_1\rangle, |q_2\rangle, |q_3'\rangle$.

$|q_0\rangle, |q_2\rangle, |q_3\rangle \xrightarrow{\text{TOFFOLY}} |q_0\rangle, |q_2\rangle, |q_3''\rangle$
$|q_0\rangle, |q_1\rangle, |q_3''\rangle \xrightarrow{\text{TOFFOLY}} |q_0\rangle, |q_1\rangle, |q_3'\rangle$

Wir fassen zusammen, wie viele Qubits und elementare Gatter die oben vorgestellten Algorithmen benötigen:

	Anzahl der Qubits		Anzahl der Gatter	
Funktion	Ein-/Ausgabe	Intern	CNOT	TOFFOLY
CARRY	4	0	1	2
CCARRY	5	0	2	2
SUM	3	0	2	0
CSUM	4	0	0	2

Tabelle 3.3: Qubit- und Gatter-Komplexität von CARRY, CCARRY, SUM und CSUM

3.6.2 Klassische Addition

Wir beschreiben nun die klassische Addition für einen Quantencomputer [VAE96]. Sie heißt klassisch, weil sie die gleichen Methoden verwendet, wie die Addition auf einem klassischen Computer.

Um die Summe von zwei n-Bit Zahlen a und b zu berechnen, addieren wir, wie bei der aus der Schule bekannten Addition, die Zahlen stellenweise auf. Eventuell entsteht dabei ein Übertrag, der im nächsten Schritt berücksichtigt werden muß. Es gibt jedoch ein Problem mit der eins-zu-eins Implementierung dieses Algorithmus auf einem Quantencomputer. Der alte Übertrag kann wegen der Reversibilität des Quantencomputers nicht mit dem neuen Überträg überschrieben werden. Deshalb müssen alle Überträge extra gespeichert werden. Dies führt dazu, daß wir $n-1$ zusätzliche Qubits für unsere Berechnungen brauchen.

Auf der nächsten Seite präsentieren wir die gesteuerte Version des Algorithmus. Bei der nicht gesteuerten Addition wird das Qubit $|\mathsf{ctrl}\rangle$ weggelassen und dessen Wert als eins angenommen.

Die Anzahl der Qubits und Gatter, die zur Ausführung der Algorithmen nötig sind, sieht wie folgt aus:

	Anzahl der Qubits		Anzahl der Gatter	
Funktion	Ein-/Ausgabe	Intern	cNOT	Toffoly
Add	$2n+1$	$n-1$	$4n$	$4n-2$
cAdd	$2n+2$	$n-1$	$2n$	$6n-1$

Tabelle 3.4: Qubit- und Gatter-Komplexität von Add und cAdd

Algorithm 8 cAdd

Input: $|\mathsf{ctrl}\rangle, |a\rangle, |b\rangle, |\mathsf{carry}_{in}\rangle$.
Output: $|\mathsf{ctrl}\rangle, |a\rangle, |b'\rangle, |\mathsf{carry}_{out}\rangle$ wie in (3.2).

1. Initialisiere $|\mathsf{carry}_0\rangle$ mit Null.
2. Für $i = 1, \ldots, n-1$ berechne bitweise den Übertrag
$$|\mathsf{carry}_{i-1}\rangle, |a_i\rangle, |b_i\rangle, |0\rangle \overset{\textsc{Carry}}{\longrightarrow} |\mathsf{carry}_{i-1}\rangle, |a_i\rangle, |b_i\rangle, |\mathsf{carry}_i\rangle$$
3. Berechne carry_n und b'_n
$$|\mathsf{ctrl}\rangle, |\mathsf{carry}_{n-1}\rangle, |a_n\rangle, |b_n\rangle, |\mathsf{carry}_{in}\rangle \overset{\textsc{cCarry}}{\longrightarrow} |\mathsf{ctrl}\rangle, |\mathsf{carry}_{n-1}\rangle, |a_n\rangle, |b_n\rangle, |\mathsf{carry}_{out}\rangle$$
$$|\mathsf{ctrl}\rangle, |a_n\rangle, |b_n\rangle \overset{\textsc{Toffoly}}{\longrightarrow} |\mathsf{ctrl}\rangle, |a_n\rangle, |b''_n\rangle$$
$$|\mathsf{ctrl}\rangle, |\mathsf{carry}_{n-1}\rangle, |a_n\rangle, |b''_n\rangle \overset{\textsc{cSum}}{\longrightarrow} |\mathsf{ctrl}\rangle, |\mathsf{carry}_{n-1}\rangle, |a_n\rangle, |b'_n\rangle$$
4. Berechne für $i = n-1, \ldots, 1$ die Summe und setze die temporären carry-Bits auf Null.
$$|\mathsf{carry}_{i-1}\rangle, |a_i\rangle, |b_i\rangle, |\mathsf{carry}_i\rangle \overset{\textsc{Carry}^\dagger}{\longrightarrow} |\mathsf{carry}_{i-1}\rangle, |a_i\rangle, |b_i\rangle, |0\rangle$$
$$|\mathsf{ctrl}\rangle, |\mathsf{carry}_{i-1}\rangle, |a_i\rangle, |b_i\rangle \overset{\textsc{cSum}}{\longrightarrow} |\mathsf{ctrl}\rangle, |\mathsf{carry}_{i-1}\rangle, |a_i\rangle, |b'_i\rangle$$

3.6.3 Quantenaddition

Um den Nachteil der klassischen Addition zu vermeiden, die $n-1$ zusätzliche Qubits benötigt, entwickelte Draper in [Dra00] einen Algorithmus, der ohne Überträge arbeitet. Um zwei Zahlen a und b zu addieren, wird zuerst $\mathrm{QFT}(b)$, dann, mit Hilfe der cR_k-Gatter, $\mathrm{QFT}(a+b)$ und schließlich $\mathrm{QFT}^\dagger(\mathrm{QFT}(a+b)) = a+b$ berechnet.

Der Vorteil dieses Algorithmus liegt darin, daß keine temporären Qubits verwendet werden. Der Nachteil liegt in der Laufzeit, die wegen der Fouriertransformation quadratisch in der Eingabelänge ist. Die Laufzeit der klassischen Addition ist dagegen linear.

Wir präsentieren nun den Algorithmus.

Algorithm 9 QADD

Input: $|a\rangle, |b\rangle, |\mathsf{carry}_{out}\rangle$.
Output: $|a\rangle, |b'\rangle$ mit $b' = a + b$.

$|b\rangle \xrightarrow{\text{QFT}} |b_{new}\rangle$
for $i = n+1, \ldots, 1$ **do**
  for $j = i, \ldots, 1$ **do**
    $|a_j\rangle, |b_i\rangle \xrightarrow{\text{CR}_{i-j+1}} |a_j\rangle, |b_{i,new}\rangle$
$|b\rangle \xrightarrow{\text{QFT}^\dagger} |b'\rangle$

Die Korrektheit des Algorithmus kann man dadurch beweisen, daß man den Zustand des Quantenregisters $|b\rangle$ verfolgt (siehe dazu [Dra00]).

Die gesteuerte Version kann so realisiert werden, daß in der **for**-Schleife das CR_k-Gatter zwischen zwei TOFFOLY-Gattern zwischengeschaltet wird. Diese TOFFOLY-Gatter haben als Steuerungsbits $|\mathsf{ctrl}\rangle$ und $|a_j\rangle$ und als Zielqubit ein zusätzliches temporäres Qubit, welches zu Beginn des Algorithmus mit 0 initialisiert werden muß.

Die Laufzeit-/Speicher-Komplexität der Quantenaddition sieht wie folgt aus:

	Anzahl der Qubits		Anzahl der Gatter		
Funktion	Ein-/Ausgabe	Intern	H	CR_k	TOFFOLY
QADD	$2n+1$	0	$2n+2$	$(n+1)(3n+2)/2$	
CQADD	$2n+2$	1	$2n+2$	$(n+1)(3n+2)/2$	$(n+1)(n+2)$

Tabelle 3.5: Qubit- und Gatter-Komplexität von QADD und CQADD

3.7 Subtraktion

Die Subtraktion ist die zu Addition inverse Operation. Deshalb kann sie dadurch implementiert werden, daß der Algorithmus für die Addition in umgekehrter Reihenfolge ausgeführt wird. Die Anzahl der Qubits und Gatter, die zur Berechnung der Subtraktion nötig sind, ist die gleiche wie bei den Additionsalgorithmen und sieht wie folgt aus:

	Anzahl der Qubits		Anzahl der Gatter			
Funktion	Ein-/Ausgabe	Intern	H	CNOT	CR_k	TOFFOLY
SUB	$2n+1$	$n-1$		$4n$		$4n-2$
CSUB	$2n+2$	$n-1$		$2n$		$6n-1$
QSUB	$2n+1$	0	$2n+2$		$(n+1)(3n+2)/2$	
CQSUB	$2n+2$	1	$2n+2$		$(n+1)(3n+2)/2$	$(n+1)(n+2)$

Tabelle 3.6: Qubit- und Gatter-Komplexität von SUB, CSUB, QSUB und CQSUB

3.8 Addition modulo einer Zahl

In diesem Abschnitt betrachten wir die doppelt gesteuerte Version der Addition modulo einer Zahl. Diesen Algorithmus werden wir für später für die gesteuerte Version der Multiplikation modulo einer Zahl verwenden. Der Algorithmus stammt aus [Bea02].

Algorithm 10 ccAddMod

Input: $|c_1\rangle, |c_2\rangle|a\rangle, |b\rangle, |N\rangle$, wobei $0 \leq a, b, < N$ gilt.
Output: $|c_1\rangle, |c_2\rangle|a\rangle, |b_{out}\rangle, |N\rangle$ mit $b_{out} = a+b \mod N$, falls $c_1 = c_2 = 1$ und $b_{out} = b$ sonst.

1. Berechne $b' = b + \mathsf{ctrl} \cdot a - N$ und c_3, wobei $c_3 = 0$, falls $b' \geq 0$, und $c_3 = 1$ sonst.

$$|c_1\rangle, |c_2\rangle, |0\rangle \xrightarrow{\text{Toffoly}} |c_1\rangle, |c_2\rangle, |\mathsf{ctrl}\rangle$$
$$|\mathsf{ctrl}\rangle, |a\rangle, |b\rangle, |0\rangle \xrightarrow{\text{cAdd}} |\mathsf{ctrl}\rangle, |a\rangle, |b + \mathsf{ctrl} \cdot a\rangle, |t_1\rangle$$
$$|b + \mathsf{ctrl} \cdot a\rangle, |N\rangle, |t_1\rangle \xrightarrow{\text{Sub}} |b'\rangle, |N\rangle, |t_2\rangle$$
$$|t_2\rangle, |0\rangle \xrightarrow{\text{cNOT}} |t_2\rangle, |c_3\rangle$$

2. Lösche c_3 und ctrl

$$|c_3\rangle, |N\rangle|b'\rangle, |t_2\rangle, \xrightarrow{\text{cAdd}} |c_3\rangle, |N\rangle|b_{out}\rangle, |0\rangle$$
$$|c_3\rangle, |a\rangle, |b_{out}\rangle, |0\rangle \xrightarrow{\text{cSub}} |c_3\rangle, |a\rangle, |b''\rangle, |t_3\rangle$$
$$|t_3\rangle, |c_3\rangle \xrightarrow[\text{cNOT}]{2\times\text{NOT}} |t_3\rangle, |0\rangle$$
$$|\mathsf{ctrl}\rangle, |a\rangle, |b''\rangle, |t_3\rangle \xrightarrow{\text{cAdd}} |\mathsf{ctrl}\rangle, |a\rangle, |b_{out}\rangle, |0\rangle$$
$$|c_1\rangle, |c_2\rangle, |\mathsf{ctrl}\rangle \xrightarrow{\text{Toffoly}} |c_1\rangle, |c_2\rangle, |0\rangle$$

3. Gebe zurück: $|c_1\rangle$, $|c_2\rangle$, $|a\rangle$, $|b_{out}\rangle$

Der Algorithmus ccAddMod hat die folgende Komplexität:

	Anzahl der Qubits	
Funktion	Ein-/Ausgabe	Intern
ccAddMod	$2n + 2$	$n + 2$
ccQAddMod	$2n + 2$	3

Tabelle 3.7: Gatter-Komplexität von ccAddMod und ccQAddMod

	Anzahl der Gatter			
Funktion	H	cNOT	cR_k	Toffoly
ccAddMod		$14n+6$		$20n$
ccQAddMod	$6n+6$	4	$(n+1)(11n+2)/2$	$4(n+1)(n+2)+2$

Tabelle 3.8: Qubit-Komplexität von ccAddMod und ccQAddMod

3.9 Vergleich

Der Vergleich kann mit Hilfe der Subtraktion implementiert werden. Um zwei ganze Zahlen a und b mit $\text{size}(a), \text{size}(b) \leq n$, zu vergleichen, wenden wir einen der oben angegebenen Subtraktionsalgorithmen auf den Zustand $|\mathsf{ctrl}\rangle, |a\rangle, |b\rangle$ an, um $a-b$ zu berechnen. Falls nach dieser Berechnung $\mathsf{carry}_{out} = 1$ ist, gilt $a < b$. Ansonsten gilt $a \geq b$. Zum Schluß wenden wir einen Additionsalgorithmus an, um die veränderten Register wiederherzustellen.

Die Anzahl der Qubits und Gatter, die zur Ausführung des Algorithmus nötig sind, sieht wie folgt aus:

	Anzahl der Qubits		Anzahl der Gatter			
Funktion	Ein-/Ausgabe	Intern	H	cNOT	cR_k	Toffoly
Cmp	$2n+1$	$n-1$		$6n+2$		$4n$
cCmp	$2n+2$	$n-1$		$6n+4$		$4n-2$
QCmp	$2n+1$	0	$4n+4$	1	$(n+1)(3n+2)$	
cQCmp	$2n+2$	0	$4n+4$		$(n+1)(3n+2)$	1

Tabelle 3.9: Qubit- und Gatter-Komplexität von Cmp, cCmp, QCmp und cQCmp

3.10 Funktionen Abs und Sgn

Um negative Zahlen darzustellen, verwenden wir die übliche Zweierkomplement-Darstellung. Deshalb lassen sich Algorithmen zur Bestimmung des Vorzeichens und des Absolutbetrags leicht implementieren.

Das positive Vorzeichen kodieren wir als eins und das negative als Null und definieren dazu die Funktion sgn:

$$\text{sgn} : \mathbb{Z} \longrightarrow \{0,1\}, \quad x \mapsto \begin{cases} 1, & \text{falls } x \geq 0 \\ 0, & \text{sonst} \end{cases}$$

Algorithm 11 Sgn

Input: $|x\rangle$.
Output: $|x\rangle$, $|s\rangle$, wobei $s = 1$, falls $x \geq 0$ ist, und $s = 0$ sonst.

$|x_n\rangle, |1\rangle \overset{\text{cNOT}}{\longrightarrow} |x_n\rangle, |s\rangle$
Gebe zurück: $|x\rangle$, $|s\rangle$

Algorithm 12 ABS

Input: $|x\rangle$.
Output: $||x|\rangle$, $|s\rangle$, wobei $s = 1$, falls $x \geq 0$ ist, und $s = 0$ sonst.

$|x_n\rangle, |1\rangle \overset{\text{cNOT}}{\longrightarrow} |x_n\rangle, |s\rangle$
$|s\rangle, |x\rangle \overset{\text{SUB}}{\longrightarrow} |s\rangle, |x - s\rangle$
for $i = 1, \ldots, n$ **do**
  $|x_i\rangle \overset{\text{cNOT}}{\longrightarrow} |x'_i\rangle$
Gebe zurück: $|x'\rangle$, $|s\rangle$

Es folgt die Anzahl der benötigten Qubits und Gatter:

	Anzahl der Qubits		Anzahl der Gatter			
Funktion	Ein-/Ausgabe	Intern	H	cNOT	cR_k	TOFFOLY
SGN	$n+1$			1		
ABS	$n+1$	$n-1$		$2n+1$		$2n-1$
QABS	$n+1$	1	$2n$	$n+1$	$n(n+2)$	

Tabelle 3.10: Qubit- und Gatter-Komplexität von SGN, ABS und QABS

3.11 Multiplikation

Als nächstes beschreiben wir zwei Multiplikationsalgorithmen, welche mit Hilfe der gesteuerten Additionen implementiert werden. Der erste Algorithmus beschreibt die folgende Quantenoperation:

$$|\mathsf{ctrl}\rangle, |a\rangle, |b\rangle, |c\rangle \overset{\text{cMULT-A}}{\longrightarrow} |\mathsf{ctrl}\rangle, |a\rangle, |b\rangle, |c'\rangle, \quad \text{mit} \quad c' = \begin{cases} c + ab, & \text{falls } \mathsf{ctrl} = 1 \\ c, & \text{falls } \mathsf{ctrl} = 0. \end{cases}$$

Die Quantenoperation, die der zweite Algorithmus beschreibt, ist

$$|\mathsf{ctrl}\rangle, |a\rangle, |b\rangle, |0\rangle \overset{\text{cMULT-B}}{\longrightarrow} |\mathsf{ctrl}\rangle, |a\rangle, |c\rangle, \quad \text{mit} \quad c = \begin{cases} ab, & \text{falls } \mathsf{ctrl} = 1 \\ b, & \text{sonst.} \end{cases}$$

Algorithm 13 cMULT-A

Input: $|\mathsf{ctrl}\rangle$, $|a\rangle$, $|b\rangle$, $|c\rangle$ mit $\operatorname{size} a, \operatorname{size} b, \operatorname{size} c \leq n$.
Output: $|\mathsf{ctrl}\rangle$, $|a\rangle$, $|b\rangle$, $|c'\rangle$ wobei $c' = c + ab$, falls $\mathsf{ctrl} = 1$, und $c' = c$ sonst.

for $i = 1, \ldots, n$ **do**
  $|\mathsf{ctrl}\rangle, |a_i\rangle, |0\rangle \overset{\text{TOFFOLY}}{\longrightarrow} |\mathsf{ctrl}\rangle, |a_i\rangle, |\mathsf{ctrl}'\rangle$
  $|\mathsf{ctrl}'\rangle, |2^{i-1}b\rangle, |c\rangle \overset{\text{cADD}}{\longrightarrow} |\mathsf{ctrl}'\rangle, |2^{i-1}b\rangle, |c_{new}\rangle$ /* $c_{new} \leftarrow c + 2^{i-1}\mathsf{ctrl}'b$ */
  $|\mathsf{ctrl}\rangle, |a_i\rangle, |\mathsf{ctrl}'\rangle \overset{\text{TOFFOLY}}{\longrightarrow} |\mathsf{ctrl}\rangle, |a_i\rangle, |0\rangle$

Algorithm 14 cMult-B

Input: $|\mathsf{ctrl}\rangle|a\rangle, |b\rangle$.

Output: $|\mathsf{ctrl}\rangle$, $|a\rangle, |b'\rangle$ mit $b' = ab$, falls $\mathsf{ctrl} = 1$, und $b' = b$ sonst.

Initialisiere $|c\rangle$ mit Null, wobei size$(c) = n$ ist.

for $i = 1, \ldots, n$ **do**

$$
\begin{aligned}
|\mathsf{ctrl}\rangle, |b_1\rangle, |0\rangle &\xrightarrow{\text{Toffoly}} |\mathsf{ctrl}\rangle, |b_1\rangle, |\mathsf{ctrl}'\rangle \\
|\mathsf{ctrl}'\rangle, |2^{i-1}a\rangle, |c\rangle, |0\rangle &\xrightarrow{\text{cAdd}} |\mathsf{ctrl}'\rangle, |2^{i-1}a\rangle, |c'\rangle, |\mathsf{carry}_{out}\rangle \\
|\mathsf{ctrl}\rangle, |b_1\rangle, |\mathsf{ctrl}'\rangle &\xrightarrow{\text{Toffoly}} |\mathsf{ctrl}\rangle, |b_1\rangle, |0\rangle \\
|\mathsf{ctrl}\rangle, |2^{i-1}a\rangle, |c\rangle, |b_1\rangle &\xrightarrow{\text{cCmp}} |\mathsf{ctrl}\rangle, |2^{i-1}a\rangle, |c\rangle, |b_1'\rangle \\
|\mathsf{ctrl}\rangle, |b_1'\rangle &\xrightarrow{\text{cNOT}} |\mathsf{ctrl}\rangle, |b_1''\rangle
\end{aligned}
$$

/* jetzt gilt $b_1'' = b_1$ falls $\mathsf{ctrl} = 0$ und $b_1'' = 0$ sonst */

Entnehme den Qubit $|b_1''\rangle$ dem $|b\rangle$ Register und füge es dem Register $|c\rangle$ hinzu.

Die Anzahl der Qubits und Gatter, die zur Ausführung der Algorithmen nötig sind, sieht wie folgt aus:

	Anzahl der Qubits		Anzahl der Gatter			
Funktion	Ein/Ausg.	Intern	H	cNOT	cR_k	Toffoly
Mult-A	$4n$	$2n-1$		$2n^2$		$6n^2 - n$
cMult-A	$4n+1$	$2n$		$2n^2$		$6n^2 + n$
QMult-A	$4n$	1	$4n$		$n^3 + 3n^2 - n$	$2n^2$
cQMult-A	$3n+1$	2	$4n$		$n^3 + 3n^2 - n$	$2n^2 + 2n$
Mult-B	$4n$	$n-1$		$8n^2 + 3n$		$10n^2 - n$
cMult-B	$4n+1$	n		$8n^2 + 3n$		$10n^2 + n$
QMult-B	$4n$	1	$4n^2 + 4n$	$2n$	$\frac{n(n+1)(7n+6)}{2}$	$2n$
cQMult-B	$4n+1$	2	$4n^2 + 4n$	n	$\frac{n(n+1)(7n+6)}{2}$	$n^3 + 3n^2 + 5n$

Tabelle 3.11: Qubit- und Gatter-Komplexität der Multiplikationsalgorithmen

3.12 Multiplikation modulo einer Zahl

Wir modifizieren den Algorithmus aus [Bea02], damit dieser auch auf negativen Zahlen operieren kann.

Der Algorithmus cMultMod ist auf der folgenden Seite abgebildet. Seine Komplexität ist:

	Anzahl der Qubits		Anzahl der Gatter			
Funktion	Ein/Ausg.	Intern	H	cNOT	cR_k	Toffoly
cMultMod	$2n+2$	$n+2$		$14n^2 + 22n$		$20n^2 + 32n$
cQMultMod	$2n+2$	3	$4n(n+6)$		$4n(n+3)^2$	$5n(n^2+5n)$

Tabelle 3.12: Qubit- und Gatter-Komplexität von cMultMod und cQMultMod

Algorithm 15 cMultMod

Input: $|\mathsf{ctrl}\rangle, |a\rangle, |b\rangle, |y\rangle, |N\rangle$, wobei $0 \leq y, |a|, |b| < N$ gilt.
Output: $|\mathsf{ctrl}\rangle, |a\rangle, |b\rangle, |y + \mathsf{ctrl} \cdot ab \mod N\rangle$.

1. Damit ccAddMod verwendet werden kann, müssen alle Parameter größer oder gleich Null sein. Berechne $a', b' > 0$ mit $a' \equiv a \mod N$ und $b' \equiv b \mod N$

$$|a\rangle, |0\rangle \xrightarrow{\text{Abs}} ||a|\rangle, |\operatorname{sgn}(a)\rangle$$
$$|\operatorname{sgn}(a)\rangle, ||a|\rangle, |N\rangle \xrightarrow[\text{cSub}]{\operatorname{sgn}(a)=0} |\operatorname{sgn}(a)\rangle, |\underbrace{N - (1 - \operatorname{sgn}(a))|a|}_{=a'}\rangle, |N\rangle$$
$$|b\rangle, |0\rangle \xrightarrow{\text{Abs}} ||b|\rangle, |\operatorname{sgn}(b)\rangle$$
$$|\operatorname{sgn}(b)\rangle, ||b|\rangle, |N\rangle \xrightarrow[\text{cSub}]{\operatorname{sgn}(b)=0} |\operatorname{sgn}(b)\rangle, |\underbrace{N - (1 - \operatorname{sgn}(b))|b|}_{=b'}\rangle, |N\rangle$$

2. Wiederhole n mal

$$|\mathsf{ctrl}\rangle, |a'_i\rangle, |2^{i-1}b'\rangle, |y\rangle, |N\rangle \xrightarrow{\text{ccAddMod}} |\mathsf{ctrl}\rangle, |a'_i\rangle, |2^{i-1}b'\rangle, |y_{new}\rangle, |N\rangle$$

   /* Verwende Register $|y_{new}\rangle$ als $|y\rangle$ in der nächsten Iteration */

3. Rekonstruiere a und b

$$|\operatorname{sgn}(b)\rangle, |b'\rangle, |N\rangle \xrightarrow[\text{cAdd}]{\operatorname{sgn}(b)=0} |\operatorname{sgn}(b)\rangle, ||b|\rangle, |N\rangle$$
$$||b|\rangle, |\operatorname{sgn} b\rangle \xrightarrow{\text{Abs}} |b\rangle, |0\rangle$$
$$|\operatorname{sgn}(a)\rangle, |a'\rangle, |N\rangle \xrightarrow[\text{cAdd}]{\operatorname{sgn}(a)=0} |\operatorname{sgn}(a)\rangle, ||a|\rangle, |N\rangle$$
$$||a|\rangle, |\operatorname{sgn}\rangle(a) \xrightarrow{\text{Abs}} |a\rangle, |0\rangle$$

4. Gebe zurück: $|\mathsf{ctrl}\rangle, |a\rangle, |b\rangle, |y\rangle$

3.13 Division

Die Division ist die zur Multiplikation inverse Operation und kann dadurch implementiert werden, daß die Operationen aus dem Multiplikationsalgorithmus in umgekehrter Reihenfolge ausgeführt werden. Die Anzahl der Qubits und Gatter ist die gleiche wie bei der Multiplikation.

3.14 Größter gemeinsamer Teiler

In diesem Abschnitt stellen wir einen Algorithmus zur Berechnung des größten gemeinsamen Teilers zweier ganzer n-Bit-Zahlen a und b vor. Das Ziel dabei ist, wie im klassischen Computermodell, einen Algorithmus zu entwickeln, welcher in Zeit $O(n^2)$ ausgeführt werden kann. Diese Zeitkomplexität basiert auf der Tatsache, daß die Anzahl der Divisionen mit Rest, die im Algorithmus ausgeführt werden, und die Komplexität dieser Divisionen zusammenhängen, und zwar je mehr Divisionen desto kleiner die Komplexität der einzelnen Divisionen und umgekehrt. Diese Tatsache läßt sich allerdings nicht eins zu eins auf Quantencomputer übertragen, da die Quantenregister nicht nur einen sondern mehrere Werte gleichzeitig enthalten können. Deshalb kann der Fall eintreten, daß für den einen Wert die Anzahl der Divisionen während für einen anderen Wert die Komplexität der Divisionen groß ist. Daraus folgt, daß der Algorithmus immer die maximale Anzahl der Divisionen mit Rest zweier n-Bit-Zahlen durchführen muß. Die Zeitkomplexität des kompletten Algorithmus ist in diesem Fall $O(n^3)$.

Um die gewünschte Zeitkomplexität trotzdem zu erreichen, modifizieren wir den Algorithmus aus [PZ03], welcher das Inverse einer Zahl modulo einer Primzahl berechnet. Wir desynchronisieren die Berechnung für jedes Paar (a, b) in der Superposition. Wir implementieren den Algorithmus als einen endlichen Automaten mit sechs Zuständen. Die Berechnung startet im Zustand A und endet im Zustand F. Wir verwenden die folgenden Zustände:

Zustand A: Finde i mit $2^{i-1} \leq a/b < 2^i$ und berechne $b \leftarrow 2^i b$
Anfangsbedingung: $i = 0$

$$|b\rangle, |i\rangle \longrightarrow |2b\rangle, |i+1\rangle$$

Endbedingung: $b > a$

Zustand B: Berechne $q = \lfloor a/b \rfloor$ und $a \leftarrow a - qb = a \mod b$ und setze $i \leftarrow i - 1$
Anfangsbedingung: $q = 0$

$$|a\rangle, |2b\rangle, |i+1\rangle, |q\rangle \longrightarrow |a - q'b\rangle, |b\rangle, |i\rangle, |2q + q'\rangle,$$

wobei q' das niederwertigste Bit von q ist.
Endbedingung: $i = 0$

Zustand C: Berechne $x_1 \leftarrow x_1 + qx_2$ und lösche q
Anfangsbedingung: $j = 0$

$$|x_1\rangle, |x_2\rangle, |2q + q'\rangle, |j\rangle \longrightarrow |x_1 + q'x_2\rangle, |2x_2\rangle, |q\rangle, |j+1\rangle$$

Endbedingung: $q = 0$

Zustand D: Lösche j und berechne den Wert von x_2 vor dem Zustand C. Anfangsbedingung: $x_2 > x_1$

$$|2x_2\rangle, |j+1\rangle \longrightarrow |x_2\rangle, |j\rangle$$

Endbedingung: $j = 0$

Zustand E: Tausche a und b, x_1 und x_2, y_1 und y_2 und ändere das Vorzeichen.

Zustand F: Erhöhe *counter* um eins.
Anfangsbedingung: $b = 0$

$$|counter\rangle \longrightarrow |counter + 1\rangle$$

Endbedingung: keine

Bei der Implementierung des Algorithmus verwenden wir vier zusätzliche Zustände, die einen einfacheren Zustandsübergang ermöglichen. Wir präsentieren nun den Algorithmus, den wir, um bessere Lesbarkeit zu erreichen, in drei Teile unterteilen.

Algorithm 16 XGCD

Input: $|a\rangle$, $|b\rangle$, wobei a und b natürliche Zahlen mit $\text{size}(a) = \text{size}\, b \leq n$ sind.
Output: $|a\rangle$, $|b\rangle$, $|g\rangle$, $|x\rangle$, $|\mathsf{temp}\rangle$ mit $ax + by = \gcd(a, b)$.

/* Initialisiere Quantenregister */
$a' \leftarrow a$, $b' \leftarrow b$
$x \leftarrow 1$, $y \leftarrow 0$
$\mathsf{sign} \leftarrow 0$, $\mathsf{state} \leftarrow 0$, $\mathsf{counter} \leftarrow 0$
$\mathsf{loop}_1 \leftarrow 0$, $\mathsf{loop}_2 \leftarrow 0$, $\mathsf{loop}_3 \leftarrow 0$, $\mathsf{loop}_4 \leftarrow 0$, $\mathsf{loop}_5 \leftarrow 0$

/* Berechne ggT */
if $b = 0$ **then** $|state\rangle \xrightarrow{\text{Add}} |state + 10\rangle$ /* Falls b=0 dann gehe in den Endzustand */
for $k \leftarrow 1$ **to** $\lceil 4.5n \rceil$ **do** XGCD-LOOP-1, XGCD-LOOP-2
if $\mathsf{sign} = 1$ **then** $|x\rangle \longrightarrow |-x\rangle$

/* Lösche y */
$|y\rangle, |b'\rangle, |a\rangle \xrightarrow{\text{Mult}} |ya\rangle, |b'\rangle, |a\rangle \xrightarrow{\text{Sub}} |0\rangle, |b'\rangle, |a\rangle$ /* Es gilt $ya = b'$ */

Gebe zurück: $|a'\rangle$, $|b'\rangle$, $|a\rangle$, $|x\rangle$, $|\mathsf{counter}, \mathsf{sign}\rangle$

Um die Korrektheit des Algorithmus XGCD zu beweisen benötigen wir das folgende Lemma.

Lemma 3.14.1 *Seien $a, b \in \mathbb{Z}$ und $a \geq b \geq 1$. Bezeichne mit $a^{(m)}$ bzw. $b^{(m)}$ a bzw. b während der m-ten Ausführung des Zustands 0. Außerdem bezeichne mit N, wie oft der Zustand 0 für gegebene a und b erreicht wird. Sei $n = \text{size}(a)$, dann gilt*

$$\sum_{k=1}^{N} (4\lfloor \log_2(a^{(k)}/b^{(k)}) \rfloor + 1) \leq 4.5n.$$

Algorithm 17 XGCD-LOOP-1

Zustand 0: Setze $\mathsf{loop}_1 \leftarrow 1$. $\quad |\mathsf{state}\rangle, |\mathsf{loop}_1\rangle \xrightarrow[\textsc{cNot}]{\mathsf{state}=0} |\mathsf{state}\rangle, |\mathsf{loop}_1'\rangle$

Übergang $0 \to 1$**:** $\quad |\mathsf{loop}_1\rangle, |\mathsf{counter}\rangle, |\mathsf{state}\rangle \xrightarrow[\textsc{Toffoly}]{\mathsf{loop}_1=1,\ \mathsf{counter}=0} |\mathsf{loop}_1\rangle, |\mathsf{counter}\rangle, |1\rangle$

Zustand 1: (Zustand A) $\mathsf{counter}' \leftarrow \mathsf{counter} + 1$, $b' \leftarrow 2b$

$$|\mathsf{state}\rangle, |\mathsf{counter}\rangle, |b\rangle \xrightarrow[\textsc{cAdd},\textsc{cLeftShift}]{\mathsf{state}=1} |\mathsf{state}\rangle, |\mathsf{counter}'\rangle, |b'\rangle$$

Übergang $1 \to 2$**:** $\quad |\mathsf{loop}_1\rangle, |a\rangle, |b\rangle, |\mathsf{state}\rangle \xrightarrow[2\times\textsc{Toffoly}]{\mathsf{loop}_1=1,\ b>a} |\mathsf{loop}_1\rangle, |a\rangle, |b\rangle, |2\rangle$

Zustand 2: Setze $\mathsf{loop}_1 \leftarrow 0$, $\mathsf{loop}_2 \leftarrow 1$.

$$|\mathsf{state}\rangle, |\mathsf{loop}_1\rangle, |\mathsf{loop}_2\rangle \xrightarrow[2\times\textsc{cNot}]{\mathsf{state}=2} |\mathsf{state}\rangle, |\mathsf{loop}_1'\rangle, |\mathsf{loop}_2'\rangle$$

Übergang $2 \to 3$**:** $\quad |\mathsf{loop}_2\rangle, |q\rangle, |\mathsf{state}\rangle \xrightarrow[\textsc{Toffoly}]{\mathsf{loop}_2=1,\ q=0} |\mathsf{loop}_2\rangle, |q\rangle, |3\rangle$

Zustand 3: (Zustand B) Falls $b \leq a$, dann setze $q' = 1$, ansonsten setze $q' = 0$. Setze $\mathsf{counter}' \leftarrow \mathsf{counter} - 1$, $q' \leftarrow 2q + q'$, $b' \leftarrow b/2$, $a' \leftarrow a - q'b'$

$$|\mathsf{state}\rangle, |\mathsf{counter}\rangle, |a\rangle, |b\rangle, |q\rangle \xrightarrow[\textsc{cRightShift},\ \textsc{cLeftShift}]{\mathsf{state}=3,\ 2\times\textsc{cSub}} |\mathsf{state}\rangle, |\mathsf{counter}'\rangle, |a'\rangle, |b'\rangle, |q'\rangle$$

Übergang $3 \to 4$**:** $\quad |\mathsf{loop}_2\rangle, |\mathsf{counter}\rangle, |\mathsf{state}\rangle \xrightarrow[3\times\textsc{Toffoly}]{\mathsf{loop}_2=1,\ \mathsf{counter}=0} |\mathsf{loop}_2\rangle, |\mathsf{counter}\rangle, |4\rangle$

Zustand 4: Setze $\mathsf{loop}_2 \leftarrow 0$, $\mathsf{loop}_3 \leftarrow 1$.

$$|\mathsf{state}\rangle, |\mathsf{loop}_2\rangle, |\mathsf{loop}_3\rangle \xrightarrow[2\times\textsc{cNot}]{\mathsf{state}=4} |\mathsf{state}\rangle, |\mathsf{loop}_2'\rangle, |\mathsf{loop}_3'\rangle$$

Übergang $4 \to 5$**:** $\quad |\mathsf{loop}_3\rangle, |\mathsf{counter}\rangle, |\mathsf{state}\rangle \xrightarrow[\textsc{Toffoly}]{\mathsf{loop}_3=1,\ \mathsf{counter}=0} |\mathsf{loop}_3\rangle, |\mathsf{counter}\rangle, |5\rangle$

Zustand 5: (Zustand C) Sei q' das niederwertigste Bit von q. Falls $q' = 1$, setze $x' \leftarrow x + y$. Nur in diesem Fall ist dann $x' \geq y$ und wir können q' löschen. Setze $\mathsf{counter}' \leftarrow \mathsf{counter} + 1$, $y \leftarrow 2y$ und $q'' \leftarrow (q - q')/2$

$$|\mathsf{state}\rangle, |q\rangle, |x\rangle, |y\rangle \xrightarrow[\textsc{cAdd}]{\mathsf{state}=5,\ q'=1} |\mathsf{state}\rangle, |q\rangle, |x'\rangle, |y\rangle$$

$$|\mathsf{state}\rangle, |x'\rangle, |y\rangle, |q'\rangle \xrightarrow[\textsc{cNot}]{\mathsf{state}=5,\ x'\geq y} |\mathsf{state}\rangle, |x'\rangle, |y\rangle, |0\rangle$$

$$|\mathsf{counter}\rangle, |y\rangle, |q - q'\rangle \xrightarrow[\textsc{cLeftShift},\textsc{cRightShift}]{\mathsf{state}=5,\ \textsc{cAdd}} |\mathsf{counter}'\rangle, |y'\rangle, |q''\rangle$$

Algorithm 18 XGCD-LOOP-2

Übergang $5 \to 6$**:** $|\mathsf{loop}_3\rangle, |q\rangle, |\mathsf{state}\rangle \xrightarrow[2\times\textsc{Toffoly}]{\mathsf{loop}_3=1,\ q=0} |\mathsf{loop}_3\rangle, |q\rangle, |6\rangle$

Zustand 6: Setze $\mathsf{loop}_3 \leftarrow 0$, $\mathsf{loop}_4 \leftarrow 1$.

$$|\mathsf{state}\rangle, |\mathsf{loop}_3\rangle, |\mathsf{loop}_4\rangle \xrightarrow[2\times\textsc{cNOT}]{\mathsf{state}=6} |\mathsf{state}\rangle, |\mathsf{loop}_3'\rangle, |\mathsf{loop}_4'\rangle$$

Übergang $6 \to 7$**:** $|\mathsf{loop}_4\rangle, |x\rangle, |y\rangle, |\mathsf{state}\rangle \xrightarrow[\textsc{Toffoly}]{\mathsf{loop}_4=1,\ x<y} |\mathsf{loop}_4\rangle, |x\rangle, |y\rangle, |7\rangle$

Zustand 7: (Zustand D) Setze $\mathsf{counter}' \leftarrow \mathsf{counter} - 1$, $y' \leftarrow y/2$

$$|\mathsf{state}\rangle, |\mathsf{counter}\rangle, |y\rangle \xrightarrow[\textsc{cSub},\ \textsc{cRightShift}]{\mathsf{state}=7} |\mathsf{state}\rangle, |\mathsf{counter}'\rangle, |y'\rangle$$

Übergang $7 \to 8$**:** $|\mathsf{loop}_4\rangle, |\mathsf{counter}\rangle, |\mathsf{state}\rangle \xrightarrow[4\times\textsc{Toffoly}]{\mathsf{loop}_4=1,\ \mathsf{counter}=0} |\mathsf{loop}_4\rangle, |\mathsf{counter}\rangle, |8\rangle$

Zustand 8: (Zustand E) Setze $\mathsf{loop}_4 \leftarrow 0$, $\mathsf{sign}' \leftarrow \mathsf{sign} \oplus 1$. Vertausche a mit b und x mit y

$$|\mathsf{state}\rangle, |\mathsf{loop}_4\rangle, |\mathsf{sign}\rangle, |a\rangle, |b\rangle, |x\rangle, |y\rangle \xrightarrow[2\times\textsc{cSwap}]{\mathsf{state}=8,\ 2\times\textsc{cNOT}} |\mathsf{state}\rangle, |\mathsf{loop}_4'\rangle, |\mathsf{sign}'\rangle, |b\rangle, |a\rangle, |y\rangle, |x\rangle$$

Übergang $8 \to 0$**:** $|\mathsf{loop}_1, \ldots, \mathsf{loop}_5\rangle, |b\rangle, |\mathsf{state}\rangle \xrightarrow[b>0,5\times\textsc{Toffoly}]{\mathsf{loop}_1=0,\ldots,\mathsf{loop}_5=0} |\mathsf{loop}_1, \ldots, \mathsf{loop}_5\rangle, |b\rangle, |0\rangle$

Übergang $8 \to 9$**:** $|\mathsf{loop}_1, \ldots, \mathsf{loop}_5\rangle, |b\rangle, |\mathsf{state}\rangle \xrightarrow[b=0,\ \textsc{Toffoly}]{\mathsf{loop}_1=0,\ldots,\mathsf{loop}_5=0} |\mathsf{loop}_1, \ldots, \mathsf{loop}_5\rangle, |b\rangle, |9\rangle$

Zustand 9: Setze $\mathsf{loop}_4 \leftarrow 0$.

$$|\mathsf{state}\rangle, |\mathsf{loop}_5\rangle \xrightarrow[\textsc{cNOT}]{\mathsf{state}=9} |\mathsf{state}\rangle, |\mathsf{loop}_5'\rangle$$

Übergang $9 \to 10$**:** $|\mathsf{loop}_5\rangle, |\mathsf{counter}\rangle, |\mathsf{state}\rangle \xrightarrow[2\times\textsc{Toffoly}]{\mathsf{loop}_5=1,\mathsf{counter}=0} |\mathsf{loop}_5\rangle, |\mathsf{counter}\rangle, |10\rangle$

Zustand 10: (Zustand F) Erhöhe counter.

$$|\mathsf{state}\rangle, |\mathsf{counter}\rangle \xrightarrow[\textsc{cAdd}]{\mathsf{state}=10} |\mathsf{state}\rangle, |\mathsf{counter}'\rangle$$

Beweis. In [PZ03], Anhang B, wird bewiesen, daß für teilerfremde a und b, wobei $a > b \geq 1$ und $a > 2$ gilt, die folgende Ungleichung wahr ist:

$$t = \sum_{k=1}^{N} (4\lfloor \log_2(a^{(k)}/b^{(k)}) \rfloor + 1) \leq 4.5 \log(a). \tag{3.3}$$

Für $a = 2$ gilt $t \leq 4.5 \log(a) + 1/2$.

Wir nehmen nun an, daß a und b nicht teilerfremd sind. Dann ist die Folge der Quotienten $\lfloor a^{(k)}/b^{(k)} \rfloor$, $k = 1, \ldots, N$, gleich der Folge der Quotienten, die bei Ausführung des Algorithmus mit Startwerten $a/\gcd(a,b)$ und $b/\gcd(a,b)$ entstehen, wobei $a/\gcd(a,b)$ und $b/\gcd(a,b)$ teilerfremd sind. Daraus folgt, daß die Gleichung (3.3) für alle a, b mit $a > b \geq 1$ gilt.

Aus der Tatsache, daß $n = \text{size}(a) > \log(a)$ und t eine ganze Zahl ist, folgt sofort die Aussage des Lemmas. □

Wir beweisen nun die Korrektheit des Algorithmus XGCD und geben obere Schranken für die Anzahl der Qubits und Gatter.

Satz 3.14.2 *Seien $n, a, b \in \mathbb{N}$, so daß $n \geq 16$ und $\text{size}(a), \text{size}(b) \leq n$. Der Algorithmus* XGCD *beschreibt die folgende Quantenoperation:*

$$|a\rangle, |b\rangle, |0\rangle, |0\rangle, |0\rangle \overset{\text{XGCD}}{\longrightarrow} |a\rangle, |b\rangle, |x\rangle, |g\rangle, |\mathsf{temp}\rangle.$$

Dabei gilt $x \in \mathbb{Z}$, $g, \mathsf{temp} \in \mathbb{N}$ mit $\text{size}(x), \text{size}(g) \leq n$, $\text{size}(\mathsf{temp}) \leq \log n + 3$ und es existiert ein $y \in \mathbb{Z}$, so daß

$$ax + by = g = \gcd(a, b).$$

Der Algorithmus benötigt $4n + \log n + 3$ Qubits für die Ein-/Ausgabe. Außerdem benötigt der Algorithmus für die internen Berechnungen $3n + 14$ Qubits, falls die Quantenaddition verwendet wird, bzw. $4n + 13$ Qubits, falls die klassische Addition verwendet wird.

Falls die Quantenaddition verwendet wird, benötigt der Algorithmus höchstens $88n^2 + 64n$ H*-Gatter, $97n^2 + 264n$* cNOT*-Gatter, $57n^3 + 162n^2 + 100n$* cR_k*-Gatter und $9n^3 + 130n^2 + 494n$* Toffoly*-Gatter. Falls die klassische Addition verwendet wird, kann der Algorithmus mit höchstens $215n^2 + 344n$* cNOT*-Gattern und $213n^2 + 336n$* Toffoly*-Gattern implementiert werden.*

Beweis. Wir beweisen zuerst die Korrektheit des Algorithmus. Wir reduzieren sie auf die Korrektheit eines klassischen euklidischen Algorithmus, dessen Korrektheit allgemein bekannt ist.

Wenn $\mathsf{state} = 8$, d.h. der Algorithmus befinden sich im Zustand E, dann wurden $a \leftarrow a - qb$, $x \leftarrow x + qy$ mit $q = \lfloor a/b \rfloor$ berechnet. Wir schätzen nun ab, wie viele k-Iterationen nötig sind, um vom Zustand 0 in den Zustand 8 zu gelangen. Im Zustand 1 befindet sich der Algorithmus solange $a \leq 2^i b$ gilt, d.h. die Anzahl der benötigten k-Iterationen ist $\lfloor a/b \rfloor + 1$. Es ist leicht einzusehen, daß die Zustände 3, 5 und 7 genauso viele k-Iterationen benötigen. Dabei wird die erste Iteration im Zustand 3 in der gleichen k-Iteration ausgeführt wie die letzte Iteration im Zustand 1. Diese Überlegung läßt sich auch auf die Zustände 5 und 7 übertragen. Es folgt also, daß die Anzahl der k-Iterationen, die benötigt werden um vom Zustand 0 in den

Zustand 8 zu gelangen, gleich $4\lfloor a/b \rfloor + 1$ ist. Aus Lemma 3.14.1 folgt, daß für gegebene a und b die Anzahl der k-Iterationen, die benötigt werden, um den größten gemeinsamen Teiler von a und b zu berechnen (d.h. in den Zustand 9 zu wechseln), kleiner als $\lceil 4, 5n \rceil$ ist.

Als nächstes schätzen wir die Anzahl der Qubits, die für die Ausführung von XGCD benötigt werden. Für die Ein-/Ausgabe werden size(a)+size(b)+size(x)+size(g)+size(counter) $= 4n + \log_2 n + 4$ Qubits benötigt. Zusätzlich werden Qubits für x, y, q, $\mathsf{loop}_1, \ldots, \mathsf{loop}_5$, temporäre Qubits für logische Operationen (z.B. während des Zustandsübergangs vom Zustand 8 zum Zustand 9) und evtl. Carry-Qubits benötigt. Zusammen sind das $3n + 14$ Qubits, falls die Quantenaddition verwendet wird, oder $4n+13$ Qubits, falls die klassische Addition verwendet wird.

Eine obere Schranke für die Anzahl der Gatter läßt sich mit den oben präsentierten Ergebnissen einfach berechnen. □

Wir fassen die Ergebnisse in den folgenden Tabellen zusammen:

	Anzahl der Qubits	
Funktion	Ein-/Ausgabe	Intern
XGCD	$4n + \log_2 n + 4$	$4n + 13$
QXGCD	$4n + \log_2 n + 4$	$3n + 14$

Tabelle 3.13: Qubit-Komplexität von XGCD und QXGCD

	Anzahl der Gatter			
Funktion	H	cNOT	cR_k	Toffoly
XGCD		$215n^2 + 344n$		$213n^2 + 336n$
QXGCD	$88n^2 + 64n$	$97n^2 + 264n$	$57n^3 + 162n^2 + 100n$	$9n^3 + 130n^2 + 494n$

Tabelle 3.14: Gatter-Komplexität von XGCD und QXGCD

Eine andere Methode, den größten gemeinsamen Teiler zu berechnen, ist der binäre GGT-Algorithmus. Bei unserer Untersuchung dieser Methode hat sich herausgestellt, daß der binäre GGT-Algorithmus mehr temporäre Qubits für die Ausführung braucht. Aus diesem Grund betrachten wir ihn in dieser Arbeit nicht näher.

3.15 Logarithmus

In diesem Abschnitt beschreiben wir, wie man den natürlichen Logarithmus einer Zahl $x \in \mathbb{Q}$, $x > 0$, mit Präzision 2^{-q} berechnet, wobei q eine positive natürliche Zahl ist. D.h. wir beschreiben einen Algorithmus, der eine der folgenden Quantenoperationen durchführt:

$$|\mathsf{ctrl}\rangle, |x\rangle, |y\rangle \xrightarrow{\text{LnAdd}} |\mathsf{ctrl}\rangle, |x\rangle, |y + \mathsf{ctrl} \cdot z\rangle, \quad \text{bzw.} \quad |\mathsf{ctrl}\rangle, |x\rangle, |y\rangle \xrightarrow{\text{LnSub}} |\mathsf{ctrl}\rangle, |x\rangle, |y - \mathsf{ctrl} \cdot z\rangle,$$

mit $z \in \mathbb{Q}$, für das gilt

$$|z - \ln x| \leq 2^{-q}. \tag{3.4}$$

Wie bereits in der Einleitung erwähnt besteht unser Ziel darin, möglichst wenige Qubits zu verwenden, und möglichst viele Operationen klassisch durchzuführen.

Der Algorithmus, den wir hier vorstellen, basiert auf den folgenden Überlegungen. Sei $x \in \mathbb{Q}$, $x > 0$, dann existiert ein $t \in \mathbb{Z}$, so daß $2^{t-1} \leq x < 2^t$. Wir setzen $y' = x/2^t$ und $y = 1 - y'$. Es gilt $1/2 \leq y' < 1$ und $0 < y \leq 1/2$. Die Tailor Zerlegung von $\ln y'$ ist:

$$\ln y' = -\sum_{i=1}^{\infty} y^i/i.$$

Wir setzen alles zusammen und erhalten

$$\ln x = \ln 2^t y' = \ln 2^t + \ln y' = t \ln 2 + \ln y' = t \ln 2 - \sum_{i=1}^{\infty} y^i/i.$$

Damit die Bedingung (3.4) erfüllt ist, müssen wir

1. eine geeignete Approximation a_0 für $\ln 2$,
2. die Anzahl der Summanden in der unendlichen Summe und
3. geeignete Approximationen a_i, $i = 1, 2, \ldots$ dieser Summanden

bestimmen.

Sei $k \in \mathbb{N}$. Es gilt

$$\sum_{i=k}^{\infty} \frac{y^i}{i} \leq \sum_{i=k}^{\infty} \frac{1}{i2^i} \leq \frac{1}{k} \sum_{i=k}^{\infty} \frac{1}{2^i} \leq \frac{1}{k2^{k-1}}.$$

Wir setzen $k = q + 1$ und folgern daraus

$$|\ln x - (t \ln 2 - \sum_{i=1}^{q} \frac{y^i}{i})| = |\ln x - (t \ln 2 - \sum_{i=1}^{k-1} \frac{y^i}{i})| \leq |\sum_{i=k}^{\infty} \frac{y^i}{i}| \leq \frac{2^{-q}}{q+1}.$$

Wenn wir also q Approximationen a_i von y^i/i mit

$$|a_i - y^i/i| \leq \frac{2^{-q}}{(q+3)}, \quad 1 \leq i \leq q,$$

und die Approximation a_0 von $t \ln 2$ mit

$$|a_0 - t \ln 2| \leq \frac{2^{-q}}{q+3}$$

berechnen (d.h. wir brauchen eine Approximation a_0' von $\ln 2$ mit $|a_0' - \ln 2| \leq \frac{2^{-q}}{(q+3)\lceil \log_2 |t| \rceil}$, dieser Wert kann klassisch berechnet werden), dann gilt für die Summe von diesen Approximationen

$$|\ln x - \sum_{i=0}^{q} a_i| \leq |a_0 - t \ln 2| + \sum_{i=1}^{q} |a_i - y^i/i| + |\sum_{i=k}^{\infty} \frac{y^i}{i}| \leq \frac{2^{-q}}{q+3} + \frac{q2^{-q}}{q+3} + \frac{2^{-q}}{q+1} \leq 2^{-q}$$

für alle $q \geq 1$.

Als nächstes beschreiben wir, wie die Summe berechnet werden kann.

Für ein $y \in \mathbb{R}$, $0 < y \leq \frac{1}{2}$, sei

$$\lceil y \rfloor_k \in \frac{1}{2^k}\mathbb{Z}$$

eine Zahl, die dadurch entsteht, daß alle Nachkommastellen nach der k-ten Stelle abgeschnitten werden. Es gilt

$$0 \leq y - \lceil y \rfloor_k < \frac{1}{2^k}.$$

Wir definieren rekursiv

$$b_1 = \lceil y \rfloor_{k+2} \text{ und } b_j = \lceil b_1 b_{j-1} \rfloor_{k+2} \text{ für } j = 2, 3, 4, \ldots, \tag{3.5}$$

dann gilt das folgende Lemma

Lemma 3.15.1 *Es gilt*

$$|y^j - b_j| \leq \frac{1}{2^k}, \quad \textit{für alle } j \in \mathbb{N}$$

Beweis. Nach Definition ist $b_1 = y + \omega_1$ mit $|\omega_1| \leq 2^{-(k+2)}$, somit gilt die Aussage des Lemmas für $j = 1$. Für $j = 2$ gilt $b_2 = \lceil b_1 b_1 \rfloor_{k+2} = (y + \omega_1)^2 + \omega_2' = y^2 + \omega_2$ mit

$$|\omega_2| \leq |2y\omega_1| + |\omega_1^2| + |\omega_2'| \leq \frac{1}{2^{k+1}} + \frac{1}{2^{2k}} \leq \frac{1}{2^k}.$$

Für $j \geq 3$ führen wir die vollständige Induktion über j. Dabei beweisen wir die folgende Aussage

$$b_j = y^j + \omega_j \quad \text{mit } |\omega_j| \leq \frac{1}{2^{k+1}} + \frac{1}{2^{k+4}} + \frac{1}{2^{2k}}. \tag{3.6}$$

Aus (3.6) folgt sofort die Aussage des Lemmas.

Für $j = 3$ gilt $b_3 = \lceil b_1 b_2 \rfloor_{k+2} = (y + \omega_1)(y^2 + \omega_2) + \omega_3' = y^3 + \omega_3$ mit

$$|\omega_3| \leq |y\omega_2| + |y^2\omega_1| + |\omega_1\omega_2| + |\omega_3'| \leq \frac{1}{2^{k+1}} + \frac{1}{2^{k+4}} + \frac{1}{2^{2k}}.$$

Wir nehmen nun an, daß die Aussage (3.6) für $j - 1$ gilt, dann gilt sie auch für j, denn

$$b_j = \lceil b_1 b_{j-1} \rfloor_{k+2} = (y + \omega_1)(y^{j-1} + \omega_{j-1}) + \omega_j' = y^j + \omega_j$$

mit

$$\begin{aligned} |\omega_j| &\leq |y\omega_{j-1}| + |y^{j-1}\omega_1| + |\omega_1\omega_{j-1}| + |\omega_j'| \\ &\leq \frac{1}{2}\left(\frac{1}{2^{k+1}} + \frac{1}{2^{k+4}} + \frac{1}{2^{2k}}\right) + \frac{1}{2^{k+j+1}} + \frac{1}{2^{2k+1}} + \frac{1}{2^{k+2}} \\ &\leq \frac{1}{2^{k+1}} + \frac{1}{2^{k+4}} + \frac{1}{2^{2k}} \end{aligned}$$

□

Als nächstes schätzen wir den Fehler ab, der bei der Division b_i/i entsteht.

Lemma 3.15.2 *Es gilt*

$$\left| \frac{y^i}{i} - \lceil \frac{b_i}{i} \rfloor_{k+2} \right| \leq \frac{1}{2^k}.$$

Beweis. Für $i = 1$ folgt die Aussage aus dem letzten Lemma. Sei nun $i \geq 2$, dann gilt

$$\left| \frac{y^i}{i} - \lceil \frac{b_i}{i} \rfloor_{k+2} \right| \leq \left| \frac{y^i}{i} - \frac{b_i}{i} \right| + \frac{1}{2^{k+2}} \leq \frac{1}{2} |y^i - b_i| + \frac{1}{2^{k+2}} \leq \frac{1}{2^{k+1}} + \frac{1}{2^{k+2}} \leq \frac{1}{2^k}.$$

□

Der Algorithmus zur Berechnung des natürlichen Logarithmus ist auf der folgenden Seite abgebildet. Seine Korrektheit folgt aus den oberen zwei Lemmas. Seine Komplexität wird im folgenden Satz untersucht.

Satz 3.15.3 *Algorithmus* LN_{ADD} *ist korrekt. Seien* q, n, m, x, y *wie im Algorithmus definiert. Setze* $p = n + m$*. Dann gelten die folgenden Aussagen.*

Zur Ein- und Ausgabe benötigt der Algorithmus $1 + \text{size}(x) + \text{size}(y)$ *Qubits. Für die internen Berechnungen sind* $m^2 + m + \log_2 p + 1$ *Qubits notwendig, falls die klassische Addition verwendet wird, bzw.* $m^2 + \log_2 p + 2$ *Qubits falls die Quantenaddition verwendet wird.*

Der Algorithmus benötigt $O(p^3)$ cNOT- *und* Toffoly-*Gatter, falls die klassische Addition verwendet wird. Falls die Quantenaddition verwendet wird, benötigt der Algorithmus* $O(p^3)$ cNOT- *und* Toffoly-*Gatter,* $O(p^2)$ H-*Gatter und* $O(p^4)$ cR_k-*Gatter.* □

Ein Algorithmus zur Berechnung von LN_{SUB} ist analog und benötigt die gleiche Anzahl der Qubits und Gatter.

3.16 Framework für Quantenalgorithmen

In diesem Abschnitt beschreiben wir kurz, wie ein typisches Quantenprogramm zur Lösung des Ordnungsproblems (mit dessen Hilfe das Faktorisieren möglich ist), des DL-Problems oder allgemein des Problems der Bestimmung des Periodengitters einer Funktion aufgebaut ist. In [Sho94] wurde das Framework für das Faktorisierungs- und das DL-Problem präsentiert. In [Kit96] wurde ein ähnliches Verfahren vorgestellt. Der Algorithmus von Kitaev verwendet jedoch die inverse Quantenfouriertransformation. Eine Verallgemeinerung des Faktorisierungs- und DL-Problem ist das Hidden-Subgroup-Problem. Algorithmen zu dessen Lösung wurden in [Hø99] und [ME99] untersucht (siehe hierzu auch [NC00]).

Als erstes muß eine effizient berechenbare periodische Funktion gefunden werden, aus deren Periode sich die Lösung des Problems ableiten läßt. Eine solche Funktion für das Ordnungsproblem ist die Funktion $f_{ord} : \mathbb{Z} \to \mathcal{G}, x \mapsto \mathsf{g}^x$, wobei g ein Element einer endlichen abelschen Gruppe $\mathcal{G}$ ist. Für das DL-Problem ist es die Funktion $f_{DL} : \mathbb{Z} \times \mathbb{Z} \to \mathcal{G}, (x, y) \mapsto \mathsf{g}^x \mathsf{h}^y$. Angenommen $\mathsf{g}^n = \mathsf{h}$, dann ist $(n, -1)$ die Periode der Funktion f_{DL}.

Wir beschreiben kurz die Idee des Frameworks. Um das Periodengitter L einer Funtkion $f : \mathbb{Z}^n \to \mathcal{S}$, mit einer Menge $\mathcal{S}$, zu berechnen, wird als erstes eine Superposition aller möglichen Werte in den ersten n Registern erzeugt und der Funktionenwert im $(n + 1)$-sten Register

Algorithm 19 LN_{ADD}

Input: $q,\ n,\ m,\ |\mathsf{ctrl}\rangle, |x\rangle,\ |y\rangle$ mit $q, n, m \in \mathbb{N}$, $x, y \in \mathbb{Q}_+$, $\text{size}(\lfloor x \rfloor), \text{size}(\lfloor y \rfloor) \leq n$, $\text{size}(x), \text{size}(y) \leq n + m$ und $q + \lceil \log_2(q+3) \rceil + \max\{2, \lceil \log_2(n+m) \rceil\} \leq m$.
Output: $|\mathsf{ctrl}\rangle, |x\rangle, |y + z\rangle$ mit $|z - \ln x| \leq 2^{-q}$.

1. Berechne t
   (a) Initialisiere $|\mathsf{state}\rangle$ und $|t\rangle$. Setze $\mathsf{state} = 0$, $t = 0$
   (b) Für $i = n-1, \ldots, -m$ führe aus

$$|x_i\rangle, |t\rangle, |\mathsf{state}\rangle \xrightarrow[\text{TOFFOLY}]{x_i=1,\ t=0} |x_i\rangle, |t\rangle, |\mathsf{state}_{new}\rangle$$
$$|\mathsf{state}\rangle, |t\rangle \xrightarrow[\text{CSUB}]{\mathsf{state}=0} |\mathsf{state}\rangle, |t_{new}\rangle$$

   (c) Setze $t \leftarrow t + n + m$: $|t\rangle \xrightarrow{\text{ADD}} |t + n + m\rangle$.

2. Berechne klassisch eine Approximation a von $\ln 2$, so daß gilt $|a - \ln 2| \leq \frac{2^{-q}}{(q+3)\lceil \log_2(n+m) \rceil}$ und setze $y \leftarrow y + \mathsf{ctrl} \cdot ta$. D.h. für $i = 0, \ldots, \text{size}(n+m) - 1$ berechne

$$a, |\mathsf{ctrl}\rangle, |t_i\rangle, |y\rangle \xrightarrow[\text{TOFFOLY,CADD}]{t_i=1,\mathsf{ctrl}=1} a, |\mathsf{ctrl}\rangle, |t_i\rangle, |y + \mathsf{ctrl} \cdot t_i \lceil 2^{-m+i} a \rfloor_{q+\lceil \log_2(q+3) \rceil + \lceil \log_2(n+m) \rceil}\rangle.$$

3. Für $i = 1, \ldots, q$ berechne b_i wie in (3.5) definiert
   (a) Setze $b_1 \leftarrow \lceil 1 - x/2^t \rfloor_{q+2+\lceil \log_2(q+3) \rceil}$.
      i. Initialisiere t: $|t\rangle \xrightarrow{\text{SUB}} |t - n - m\rangle$
      ii. Wiederhole $n + m$ mal für $i = n-1, \ldots, -m$

$$|t\rangle, |x\rangle, |0\rangle \xrightarrow[\text{CMULT}]{t<0} |t\rangle, |2x\rangle$$
$$|t\rangle \xrightarrow{\text{ADD}} |t + 1\rangle$$

      iii. Kopiere: $|x\rangle, |0\rangle \xrightarrow{\text{COPY}} |x\rangle, |b_1\rangle$.
   (b) Wiederhole für $i = 2, \ldots, q$

$$|b_1\rangle, |b_{i-1}\rangle, |0\rangle \xrightarrow{\text{MULT}} |b_1\rangle, |b_{i-1}\rangle, |\lceil b_1 b_{i-1} \rfloor_{q+2+\lceil \log_2(q+3) \rceil}\rangle$$

4. Berechne $y \leftarrow y - \sum_{i=1}^{q} \lceil b_i / i \rfloor_{q+2+\lceil \log_2(q+3) \rceil}$. Für $i = 1, \ldots, q$ führe aus

$$|\mathsf{ctrl}\rangle, |b_i\rangle, |0\rangle, |y\rangle \xrightarrow{\text{DIV}} |\mathsf{ctrl}\rangle, |\mathsf{temp}\rangle, |\lceil b_i/i \rfloor_{q+2+\lceil \log_2(q+3) \rceil}\rangle, |y\rangle \xrightarrow[\text{CSUB}]{\mathsf{ctrl}=1}$$
$$|\mathsf{ctrl}\rangle, |\mathsf{temp}\rangle, |\lceil b_i/i \rfloor_{q+2+\lceil \log_2(q+3) \rceil}\rangle, |y - \mathsf{ctrl} \cdot \lceil b_i/i \rfloor_{q+2+\lceil \log_2(q+3) \rceil}\rangle \xrightarrow{\text{MULT}}$$
$$|b_i\rangle, |0\rangle, |y + \lceil b_i/i \rfloor_{q+2+\lceil \log_2(q+3) \rceil}\rangle$$

5. Führe die Schritte 3 und 1 rückwärts aus, um temporäre Register zu löschen
6. Gebe zurück: $|\mathsf{ctrl}\rangle$, $|x\rangle$, $|y\rangle$

ausgerechnet. Nach einer Messung des letzten Registers bleiben in den ersten n Registern nur Werte der Form $\mathbf{x}' + L$, wobei $\mathbf{x}'$ ein zufälliger Vektor ist. Die Quantenfouriertransformation verändert diesen Zustand in den Zustand

$$\sum_{\mathbf{y}} \sum_{k \in \mathcal{L}} e^{2\pi i(\mathbf{x}'+\mathbf{k})\cdot \mathbf{y}/q'} |\mathbf{y}\rangle |f(\mathbf{x}')\rangle.$$

Die Wahrscheinlichkeit einen Wert $\mathbf{y}$ zu messen ist

$$\left| \sum_{k \in \mathcal{L}} e^{2\pi i(\mathbf{x}'+\mathbf{k})\cdot \mathbf{y}/q} \right|^2 = \left| \sum_{k \in \mathcal{L}} e^{2\pi i(\mathbf{k}\cdot \frac{1}{q}\mathbf{y})} \right|^2.$$

Falls nun $\frac{1}{q}\mathbf{y}$ eine Approximation eines Vektors aus L^* ist, dann liegt das innere Produkt $p = (\mathbf{k} \cdot \mathbf{y})/p$ "nahe" an einer ganzen Zahl. Deshalb wird die Summe $\sum_{k \in \mathcal{L}} e^{2\pi i p}$ groß. Falls $\mathbf{y} \notin L$ gilt, dann ist das Produkt $p = (\mathbf{k} \cdot \mathbf{y})/q$ keine ganze Zahl und es liegt auch nicht "nah" daran, so daß die Summanden $e^{2\pi i p}$ sich gegenseitig auslöschen und die Summe $\sum_{k \in \mathcal{L}} e^{2\pi i p}$ klein wird. Es folgt also, daß mit großer Wahrscheinlichkeit eine Approximation eines Vektors aus L^* gemessen wird. Durch Wiederholungen des oben genannten Algorithmus bekommen wir Approximationen von Vektoren aus L^*. Mit dem Algorithmus aus dem zweiten Kapitel (Siehe Satz 2.4.24) kann daraus eine Basis des gesuchten Gitters L berechnet werden.

Das Framework ist auf der folgenden Seite angegeben.

3.17 Semi-klassische Quantenfouriertransformation

Um die Anzahl der benötigten Qubits zu reduzieren, wurde in [GN96] die semi-klassiche Quantenfouriertransformation vorgeschlagen. Die genaue Analyse des Frameworks aus dem letzten Abschnitt zeigt, daß das Messen im Schritt 1d mit den Schritten 1e und 1f vertauscht werden kann. Außerdem ist es möglich die Erzeugung der Superposition, die Funktionsberechnung, die Quantenfouriertransformation und die abschließende Messung nicht nur in Blöcken, wie im Algorithmus dargestellt, anzuordnen, sondern auch diese Schritte für jedes einzelne Bit von $x_1, \ldots, x_m$ nacheinander auszuführen. Durch diese Umordnung wird erreicht, daß für die Speicherung von $x_1, \ldots, x_m$ nur ein Qubit anstelle von ursprünglich $m \log q$ Qubits benötigt wird.

Diese Prozedur läßt sich auf alle Funktionen anwenden, die mit schneller Exponentation berechnet werden können.

Algorithm 20 Framework: Ordnungsproblem, DL, Besitmmung des Periodengitters einer Funktion, ...

Input: $\mathsf{g}_1, \ldots, \mathsf{g}_m \in \mathcal{G}$.
Output: Eine Basis von L_f.

1. Wiederhole solange, bis ein Erzeugendensystem von L^* gefunden wird:
   (a) Initialisiere $\underbrace{|0\rangle, \ldots, |0\rangle}_{m \text{ mal}} |1\rangle$
   (b) Erzeuge die Superposition
   $$\sum_{x_1=0}^{q-1} \cdots \sum_{x_m=0}^{q-1} |x_1\rangle, \ldots, |x_m\rangle |1\rangle$$
   mit $q \in \mathbb{Z}$. Dieser Wert muß passend gewählt werden.
   (c) Berechne die Funktion, benutze dabei schnelle Exponentation
   $$\sum_{x_1=0}^{q-1} \cdots \sum_{x_m=0}^{q-1} |x_1\rangle, \ldots, |x_m\rangle |f(x_1, \ldots, x_m)\rangle$$
   (d) Messe den Funktionswert
   $$\sum_{\mathbf{k} \in \mathcal{L}} |\mathbf{x}' + \mathbf{k}\rangle |f(\mathbf{x}')\rangle,$$
   wobei $\mathbf{x}'$ zufällig und fast gleichverteilt aus der Menge $\{0, \ldots, q-1\}^n$ gewählt wird und $\mathcal{L} = \{\, k \in L_f \mid 0 \leq x'_i + k_i < q \text{ für } 1 \leq i \leq m \,\}$.
   (e) Wende die Quantenfouriertransformation an
   $$\sum_{y_1=0}^{q'-1} \cdots \sum_{y_m=0}^{q'-1} \sum_{k \in \mathcal{L}} e^{2\pi i (\mathbf{x}' + \mathbf{k}) \cdot \mathbf{y} / q'} |\mathbf{y}\rangle |f(\mathbf{x}')\rangle$$
   mit $q' \in \mathbb{Z}$, dessen Wert von q und m' abhängt.
   (f) Die abschließende Messung der ersten m Register liefert ein $\mathbf{y}'$, welches mit hoher Wahrscheinlichkeit eine sehr gute Approximation eines Vektors aus L^* ist.
2. Gegeben eine Approximation eines Erzeugendensystems von L^*, berechne klassisch eine Basis von L.

Kapitel 4

Algorithmen für imaginär-quadratische Zahlkörper

In diesem Kapitel werden Quantenalgorithmen zur Lösung des Ordnungs- sowie Diskreter-Logarithmus-Problems in imaginär-quadratischen Zahlkörpern präsentiert. Verglichen mit reell-quadratischen Zahlkörpern sowie Zahlkörpern eines höheren Grades ist der vorliegende Fall algorithmisch einfacher zu lösen, da die Idealklassen in imaginär-quadratischen Körpern einen eindeutigen Vertreter haben, der in klassischer Polynomzeit bestimmt werden kann.

Dieses Kapitel ist wie folgt aufgebaut: Im ersten Abschnitt wird ein Algorithmus zur Multiplikation zweier Ideale in quadratischen Zahlkörpern präsentiert. Im zweiten Abschnitt wird ein Algorithmus angegeben, der als Eingabe ein Ideal bekommt und es reduziert. Im nächsten Abschnitt wird ein Algorithmus für die Multiplikation zweier reduzierten Ideale beschrieben, dessen Ausgabe wieder ein reduziertes Ideal ist. In den letzen zwei Abschnitten werden Algorithmen zur Berechnung der Ordnung und des diskreten Logarithmus eines Ideals in der Klassengruppe präsentiert. Zu allen Algorithmen werden präzise obere Schranken für die Anzahl der Qubits und der elementaren Quantengatter angegeben, welche für die Ausführung der Algorithmen notwendig sind.

4.1 Multiplikation von Idealen in quadratischen Zahlkörpern

In diesem Abschnitt präsentieren wir einen Quantenalgorithmus, der bei Eingabe von zwei reduzierten Idealen das Produkt dieser Ideale berechnet. Wir geben obere Schranken für die Anzahl der Qubits und Gatter, die nötig sind, um diesen Algorithmus auszuführen.

Für den Algorithmus verwenden wir die Formel aus der Definition 2.5.46. Der Algorithmus ist auf der nächsten Seite abgebildet. Seine Laufzeit und Speicherverbrauch werden im folgenden Satz untersucht.

Satz 4.1.1 *Sei Δ eine Diskriminante, (a_1, b_1) und (a_2, b_2) reduzierte $\mathcal{O}_\Delta$-Ideale eines quadratischen Zahlkörpers. Der Algorithmus* Ideal-Mult *führt die folgende Quantenoperation durch:*

$$|a_1\rangle, |b_1\rangle, a_2, b_2, \Delta, |0\rangle, |0\rangle \longrightarrow |a_1\rangle, |b_1\rangle, a_2, b_2, \Delta, |a\rangle, |b\rangle,$$

Algorithm 21 IDEAL-MULT

Input: $|a_1\rangle$, $|b_1\rangle$, a_2, b_2, Δ, wobei (a_1, b_1) und (a_2, b_2) reduzierte $\mathcal{O}_\Delta$-Ideale eines quadratischen Zahlkörpers sind.
Output: $|a\rangle, |b\rangle, |a_1\rangle, |b_1\rangle$ mit $(a, b) = (a_1, b_1) * (a_2, b_2)$ und $0 \leq b < 2a$

1. Wende XGCD, XGCD† und XGCD an, um j', j'', m' und m zu berechnen, so dass gilt: $j'a_2 + k'a_1 = m' = \mathrm{ggT}(a_1, a_2)$ und $j''m' + l\frac{b_1+b_2}{2} = m = \mathrm{ggT}(m', \frac{b_1+b_2}{2})$ mit $k', l \in \mathbb{Z}$. Dann gilt

$$j'j''a_2 + k'j''a_1 + l\frac{b_1 + b_2}{2} = m = \mathrm{ggT}(a_1, a_2, \frac{b_1 + b_2}{2}). \tag{4.1}$$

2. Berechne $a = a_1a_2/m^2 = (a_1/m)(a_2/m)$

$$|a_1\rangle, |m\rangle, |0\rangle, |0\rangle \xrightarrow{\text{DIV}} |\frac{a_1}{m}\rangle, |m\rangle, |0\rangle, |0\rangle \xrightarrow{\text{DIV}} |\frac{a_1}{m}\rangle, |m\rangle, |\frac{a_2}{m}\rangle, |0\rangle \xrightarrow{\text{MULT}} |\frac{a_1}{m}\rangle, |m\rangle, |a\rangle$$

3. Berechne $t = j'j''a_2/m$ und $b = j'j''a_2b_1/m \mod 2a = tb_1 \mod 2a$

$$|j'\rangle, |0\rangle, |0\rangle, |j''\rangle, |a\rangle \xrightarrow{\text{MULT}} |j'j''\rangle, |0\rangle, |j''\rangle, |a\rangle \xrightarrow[\text{DIV}]{\text{MULT}} |j'j''\frac{a_2}{m}\rangle, |j''\rangle, |a\rangle$$
$$|t\rangle, |b_1\rangle, |a\rangle, |0\rangle \xrightarrow{\text{MULTMOD}} |t\rangle, |b_1\rangle, |a\rangle, |b\rangle$$

4. Berechne $b = b+(j''m'/m-j'j''a_2/m)b_2 \mod 2a = b+k'j''a_1b_2/m \mod 2a$ und lösche t

$$|t\rangle, |m'\rangle, |m\rangle, |j''\rangle \xrightarrow[\text{MULT}]{\text{DIV}} |j''\frac{m'}{m} - t\rangle, |\frac{m'}{m}\rangle, |m\rangle, |j''\rangle$$
$$|j''\frac{m'}{m} - t\rangle, |b\rangle \xrightarrow{\text{MULTMOD}} |j''\frac{m'}{m} - t\rangle, |b + (j''\frac{m'}{m} - t)b_2 \mod 2a\rangle$$
$$|j''\frac{m'}{m} - t\rangle, |\frac{m'}{m}\rangle, |m\rangle, |j''\rangle \xrightarrow[2\times\text{DIV}]{3\times\text{MULT}} |j'\rangle, |0\rangle, |0\rangle, |m'\rangle, |m\rangle, |j''\rangle$$

5. Berechne $l = 2(m - j''m')/(b_1 + b_2)$ und $b = b + l(b_1b_2 + \Delta)/(2m) \mod 2a$

$$|j''\rangle, |0\rangle, |m'\rangle, |m\rangle, |b_1\rangle \xrightarrow[\text{ADD}]{\text{MULT}} |j''m'\rangle, |m'\rangle, |m\rangle, |b_1 + b_2\rangle \xrightarrow[\text{SUB}]{\text{SUB, DIV}} |l\rangle, |0\rangle, |m'\rangle, |m\rangle, |b_1\rangle$$
$$|b_1\rangle, |0\rangle, |m\rangle \xrightarrow[\text{DIV}]{\text{MULT, ADD}} |(b_1b_2 + \Delta)/(2m)\rangle, |m\rangle$$
$$|\frac{b_1b_2 + \Delta}{2m}\rangle, |l\rangle, |a\rangle, |b\rangle \xrightarrow{\text{MULTMOD}} |\frac{b_1b_2 + \Delta}{2m}\rangle, |l\rangle, |a\rangle, |b + l\frac{b_1b_2 + \Delta}{2m} \mod 2a\rangle$$
$$|(b_1b_2 + \Delta)/(2m)\rangle, |a_1/m\rangle, |m\rangle \xrightarrow[\text{SUB, DIV}]{2\times\text{MULT}} |b_1\rangle, |0\rangle, |a_1\rangle, |m\rangle$$

6. Wende XGCD†,XGCD und XGCD†an, um die Register $|j'\rangle, |j''\rangle, |m'\rangle, |m\rangle$ zu löschen.

7. Gebe aus: $|a\rangle, |b\rangle, |a_1\rangle, |b_1\rangle$

wobei (a, b) das Produkt dieser Ideale ist.

Sei $n = size(\Delta)$. Der Algorithmus benötigt für die Ein- und Ausgabe $3n+3$ Qubits. Zusätzlich benötigt der Algorithmus höchstens $3,5n + \log_2 n + 20$ Qubits für die internen Berechnungen, wenn die Quantenaddition verwendet wird, bzw. $4n + \log_2 n + 20$ Qubits, wenn die klassische Addition verwendet wird.

Die Anzahl der elementaren Quantengatter, die für die Ausführung des Algorithmus notwendig sind, ist in der folgenden Tabelle zusammengefasst.

	Anzahl der Gatter			
Funktion	H	cNOT	cR_k	Toffoly
Ideal-Mult		$1059n^2 + 1178n$		$1227n^2 + 1132n$
QIdeal-Mult	$600n^2 + 840n$	$504n^2 + 160n$	$612n^3 + 1458n^2 + 813n$	$324n^3 + 1866n^2 + 3032n$

Tabelle 4.1: Gatter-Komplexität von Ideal-Mult und QIdeal-Mult

Beweis. Durch elementare Umformungen läßt sich zeigen, dass die Zahlen a und b, welche der Algorithmus zurückgibt, die Formeln aus dem Satz 2.5.46 erfüllen.

Wir schätzen nun die Anzahl der Qubits ab, die der Algorithmus für die Ein-/Ausgabe und für interne Berechnungen braucht. Sei $n = \text{size}(\Delta)$. Da nach Voraussetzung sowohl (a_1, b_1) als auch (a_2, b_2) reduziert sind, folgt aus dem Satz 2.5.56, daß $\text{size}(a_1), \text{size}(a_2), \text{size}(b_1), \text{size}(b_2) \leq n/2 + 1$ ist.

Als Eingabe bekommt der Algorithmus zwei Zahlen a_1 und b_1, die in Quantenregistern der Größe jeweils $n/2 + 1$ gespeichert werden können. Die anderen Eingabeparameter liegen klassisch vor und müssen in keinem Quantenregister gespeichert werden. Bei der Ausgabe kommen zusätzlich die Zahlen a, b dazu, wobei gilt $a \leq a_1 a_2 < |\Delta|$ und $0 \leq b < 2a$, d.h. $\text{size}(a) \leq n$ und $\text{size}(b) \leq n + 1$. Zusammen ergibt sich, daß für die Ein-/Ausgabe $3n + 3$ Qubits gebraucht werden.

Die Anzahl der internen Qubits sowie die Anzahl der Quantengatter läßt sich mit den Ergebnissen aus dem Kapitel 3 leicht berechnen. □

4.2 Reduktion von Idealen

Wir stellen nun einen Algorithmus zur Reduktion eines Ideals eines imaginär-quadratischen Zahlkörpers vor.

Wegen einer besseren Übersicht unterteilen wir den Algorithmus in drei Teile. Als erstes beschreiben wir, wie die Funktion ρ (siehe Definition 2.5.50) implementiert werden kann. Als zweites zeigen wir, wie man den Normalisierungsfaktor platzsparend speichert (dieser Faktor ist für die Reversibilität notwendig). Schließlich führen wir alles zusammen zum Reduktionsalgorithmus Reduce-IQ.

Algorithm 22 cRho-IQ

Input: $|\mathsf{ctrl}\rangle$, $|a\rangle$, $|b\rangle$, Δ, wobei (a, b) ein normales, nicht reduziertes $\mathcal{O}_\Delta$-Ideal mit $a \leq |\Delta|$ ist.

Output: $|\mathsf{ctrl}\rangle$, $|a'\rangle$, $|b'\rangle$ und $|s\rangle$, wobei $(a', b') = \rho(a, b)$ und $s = \lfloor (b + a')/(2a') \rfloor$, falls $\mathsf{ctrl} = 1$, und $(a', b') = (a, b)$ und $s = 0$, falls $\mathsf{ctrl} = 0$.

1. Berechne $a' = (b^2 - \Delta)/(4a)$

$$|a\rangle, |b\rangle, ||\Delta|\rangle \xrightarrow{\text{Mult}} |a\rangle, |b\rangle, ||\Delta| + b^2\rangle \xrightarrow{\text{Div}} |a\rangle, |b\rangle, |a'\rangle$$

2. Wenn $\mathsf{ctrl} = 0$, setze $b_{new} = -b$ und vertausche a' und a.

$$|\mathsf{ctrl}\rangle, |b\rangle \xrightarrow{\text{cSub}} |\mathsf{ctrl}\rangle, |(-1)^{1-\mathsf{ctrl}} b\rangle$$
$$|\mathsf{ctrl}\rangle, |a\rangle, |a'\rangle \xrightarrow{\text{cSwap}} |\mathsf{ctrl}\rangle, |a_{new}\rangle, |a'_{new}\rangle$$

3. Lösche a

$$|a\rangle, |a'\rangle, |b\rangle \xrightarrow{\text{Mult}} |4aa'\rangle, |a'\rangle, |b\rangle \xrightarrow{\text{Mult}} |4aa' - b^2\rangle, |a'\rangle, |b\rangle \xrightarrow{\text{Add}} |\underbrace{4aa' - b^2 + \Delta}_{=0}\rangle, |a'\rangle, |b\rangle$$

4. Berechne $s = \lfloor (b + a')/(2a') \rfloor$

$$|a'\rangle, |b\rangle, |0\rangle \xrightarrow{\text{Add}} |a'\rangle, |b + a'\rangle, |0\rangle \xrightarrow{\text{Div}} |a'\rangle, |b + a' \mod 2a'\rangle, |s\rangle$$
$$|a'\rangle, |s\rangle, |b + a' \mod 2a'\rangle \xrightarrow[\text{Sub}]{\text{Mult}} |a'\rangle, |s\rangle, |b\rangle$$

5. Berechne $b' = -b + 2sa'$

$$|a'\rangle, |b\rangle, |s\rangle \xrightarrow{\text{Sub}} |a'\rangle, |-b\rangle, |s\rangle \xrightarrow{\text{Mult}} |a'\rangle, |b'\rangle, |s\rangle$$

6. Gebe aus: $|\mathsf{ctrl}\rangle$, $|a'\rangle$, $|b'\rangle$ und $|s\rangle$

Die Qubit- und Gatter-Komplexität von cRho-IQ werden im folgenden Satz untersucht.

Satz 4.2.1 *Seien Δ eine Diskriminante eines imaginär-quadratischen Zahlkörpers und (a, b) ein normales, nicht reduziertes $\mathcal{O}_\Delta$-Ideal mit $a \leq |\Delta|/3$. Sei außerdem $n = \text{size}(\Delta)$. Der Algorithmus* cRho-IQ *führt die folgende Quantenoperation durch:*

$$|\mathsf{ctrl}\rangle, |a\rangle, |b\rangle, |0\rangle, \Delta \longrightarrow |\mathsf{ctrl}\rangle, |a'\rangle, |b'\rangle, |s\rangle, \Delta,$$

wobei $(a', b') = \rho(a, b)$ und $s = \lfloor (b + a')/(2a') \rfloor$, falls $\mathsf{ctrl} = 1$ ist, und $(a', b') = (a, b)$ und $s = 0$, falls $\mathsf{ctrl} = 0$ gilt.

Für die Ein-/Ausgabe benötigt der Algorithmus $2,5n$ Qubits. Zusätzlich sind $1,5n+1$ Qubits, wenn die Quantenaddition verwendet wird, bzw. $2,5n+2$ Qubits, wenn die klassische Addition verwendet wird, für interne Berechnungen nötig.

Die Anzahl der elementaren Quantengatter ist in der folgenden Tabelle zusammengefaßt

	Anzahl der Gatter			
Funktion	H	cNOT	cR_k	Toffoly
cRho-IQ		$36n^2 + 16n$		$52n^2 + n$
cQRho-IQ	$8n^2 + 16n$	$8n$	$16n^3 + 32n^2 + 10n$	$4n^2 + 11n$

Tabelle 4.2: Gatter-Komplexität von cRho-IQ und cQRho-IQ

Beweis. Daß der Algorithmus die Funktion ρ aus der Definition 2.5.50 berechnet ist leicht einzusehen.

Wir bestimmen nun eine obere Schranke für die Anzahl der Qubits für die Ein-/Ausgabe. Nach Voraussetzing gilt $\text{size}(b) \leq \text{size}(a) \leq \text{size}(\Delta) - 1$. Da (a, b) normal ist, gilt auch $a' \leq a$. Falls $\mathsf{ctrl} = 0$ gilt, werden die Werte a und a' nicht vertauscht. In diesem Fall wird im Schritt 4 das Urspruchsideal (a, b) normalisiert. Da nach Voraussetzung (a, b) bereits normal ist, gilt $s = 0$. Falls $\mathsf{ctrl} = 1$ gilt, werden a und a' vertauscht. Die Größe von s läßt sich wie folgt abschätzen: Seien i, p_{i-1}, p_i und p_{i+1} wie im Satz 2.5.54 definiert. Dann gilt

$$s_i p_i = p_{i+1} + p_{i-1} \leq 2a/\sqrt{|\Delta|} + 2a/\sqrt{|\Delta|} \leq 2\sqrt{|\Delta|}.$$

Daraus folgt $\text{size}(s) \leq 0,5n + 1$. Wir folgern, daß der Algorithmus für die Ein- und Ausgabe höchstens

$$\text{size}(\mathsf{ctrl}) + \text{size}(a) + \text{size}(b) + \text{size}(s) \leq 2,5n$$

Qubits benötigt.

Mit den Schranken für a, b, a' und s läßt sich die Anzahl der Gatter sowie die Anzahl der Qubits für interne Berechnungen aus den Ergebnissen des Kapitels 3 einfach herleiten. □

Als nächstes beschreiben wir einen Algorithmus zur effizienten Speicherung des Normalisierungsfaktors s. Dieser Faktor fällt jedes mal an, wenn der Algorithmus cRho-IQ ausgeführt wird und ist für die Reversibilität notwendig. Wie wir später in diesem Kapitel sehen werden, wird cRho-IQ $O(\log_2|\Delta|)$ mal ausgeführt. Wie wir im Beweis vom letzen Satz gesehen haben, gilt $\text{size}(s) = \text{size}(\Delta)/2 + 1$, so daß der naive Ansatz, jedes mal ein neues Register für s zu benutzen, zu einer in $\log(\Delta)$ quadratischen Speicherkomplexität führt.

Es gibt jedoch eine effizientere Methode, Normalisierungsfaktoren zu speichern. Sie basiert auf dem Satz 2.5.54. Anstatt jedes mal ein neues Register für s zu verwenden, berechnen wir in jeder Iteration die Zahl p, wie im Satz 2.5.54 beschrieben und speichern zusätzlich das Vorzeichen vom alten p. Im weiteren werden wir zeigen, daß $\text{size}(p) = O(\log_2|\Delta|)$ gilt, so daß wir einen reversiblen Algorithmus bekommen, welcher nur $O(\log_2|\Delta|)$ Qubits zum Speichern von allen s braucht.

Der Algorithmus cUncomputeS, der auf der folgenden Seite abgebildet ist, hat die obengenannten Eigenschaften. Er berechnet die neuen p's und löscht das Register, welches s enthält.

Algorithm 23 cUncomputeS

Input: $|\mathsf{ctrl}\rangle$, $|p\rangle$, $|q\rangle$, $|s\rangle$, wobei $|q| > |p|$ ist.
Output: $|\mathsf{ctrl}\rangle$, $|p'\rangle$, $|q'\rangle$ und $|\operatorname{sgn}(p)\rangle$ mit $p' = q$ und $q' = sq - p$, falls $\mathsf{ctrl} = 1$, und $p' = p$ und $q' = q$, falls $\mathsf{ctrl} = 0$.

1. Berechne $p' = q$ und $q' = -p + sq$

$$|p\rangle, |q\rangle, |s\rangle, |0\rangle \xrightarrow{\text{cNOT}} |p\rangle, |q\rangle, |s\rangle, |\operatorname{sgn}(p)\rangle \xrightarrow{\text{Mult}} |q'\rangle, |p'\rangle, |s\rangle, |\operatorname{sgn}(p)\rangle$$

2. Berechne s aus dem neuen p und q. Verwende dabei s_1 und s_2 mit $s_1 = 1$ genau dann, wenn $\operatorname{sign}(q') \neq \operatorname{sign}(q)$, d.h. $\operatorname{sign}(s) = -1$, und $s_2 = 1$ genau dann, wenn $\operatorname{sign}(q') \neq \operatorname{sign}(p)$

$$\begin{aligned}
|p'\rangle, |q'\rangle, |0\rangle, |0\rangle &\xrightarrow{2\times\text{Abs}} ||p'|\rangle, ||q'|\rangle, |\operatorname{sgn}(p')\rangle, |\operatorname{sgn}(q')\rangle \\
||p'|\rangle, ||q'|\rangle, |0\rangle &\xrightarrow{\text{Div}} ||p'|\rangle, ||q'| \bmod |p'|\rangle \left|\left\lfloor |q'|/|p'| \right\rfloor\right\rangle \\
|\operatorname{sgn}(q')\rangle, |\operatorname{sgn}(p')\rangle, |\operatorname{sgn}(p)\rangle &\xrightarrow{2\times\text{cNOT}} |\operatorname{sgn}(q)\rangle, |s_1\rangle, |s_2\rangle \\
|s_1\rangle, |s_2\rangle, \left|\left\lfloor |q'|/|p'| \right\rfloor\right\rangle &\xrightarrow[\text{cSub}]{\text{Mult}} |s_1\rangle, |s_2\rangle, |\underbrace{(-1)^{s_1}(\left\lfloor |q'|/|p'| \right\rfloor + s_2)}_{=s}\rangle
\end{aligned}$$

3. Falls $\mathsf{ctrl} = 1$, lösche s

$$|\mathsf{ctrl}\rangle, |s\rangle, |s\rangle \xrightarrow[\text{cCopy}]{\mathsf{ctrl}=1} |\mathsf{ctrl}\rangle, |s\rangle, |s(1 - \mathsf{ctrl})\rangle$$

4. Führe den Schritt 2 rückwärts aus, um die temporären Register zu löschen

5. Gebe aus: $|\mathsf{state}\rangle$, $|p\rangle$, $|q\rangle$, $|\operatorname{sgn} p\rangle$

Satz 4.2.2 *Seien $p, q, s \in \mathbb{Z}$, so daß* $\text{size}(p), \text{size}(q), \text{size}(p_{new}), \text{size}(q_{new}), size(s) \leq n$ *und* $|q| > |p|$ *gilt. Der Algorithmus* cUncomputeS *führt die folgende Quantenoperation durch:*

$$|\mathsf{ctrl}\rangle, |p\rangle, |q\rangle, |s\rangle, |0\rangle \longrightarrow |\mathsf{ctrl}\rangle, |p_{new}\rangle, |q_{new}\rangle, |s_{new}\rangle, |\operatorname{sgn} p\rangle,$$

wobei gilt: $p = p_{new}$, $q = q_{new}$ *und* $s = s_{new}$, *falls* $\mathsf{ctrl} = 0$, *und* $q = p_{new}$, $sq - p = q_{new}$ *und* $s_{new} = 0$, *falls* $\mathsf{ctrl} = 1$.

Der Algorithmus benötigt $3n + 2$ *Qubits für die Ein-/Ausgabe und außerdem höchstens* $n + 5$ *Qubits, falls die Quantenaddition verwendet wird, oder* $2n + 6$ *Qubits, falls die klassische Addition verwendet wird, für die internen Berechnungen.*

Eine obere Schranke für die Anzahl der Elementargatter ist in der folgenden Tabelle angegeben

	Anzahl der Gatter			
Funktion	H	cNOT	cR_k	Toffoly
cUncomputeS		$10n^2 + 8n + 5$		$30n^2 + 12n$
cQUncomputeS	$28n + 4$	$2n + 2$	$5n^3 + 20n^2 + 15n$	$12n^2 + 7n + 4$

Tabelle 4.3: Gatter-Komplexität von cUncomputeS und cQUncomputeS

□

Lemma 4.2.3 *Seien $p, q, s \in \mathbb{Z}$, so daß $|q| > |p|$ und $|s| > 1$ gilt. Bei der Eingabe von p, q und s berechnet der Algorithmus* cUncomputeS p_{new} *und* q_{new} *für die gilt:* $|q_{new}| > |p_{new}|$.

Beweis. Es gilt $|q_{new}| = |sq - p| \geq |s||q| - |p| \geq (|s| - 1)|q| \geq |q| = |p_{new}|$. □

Wir führen nun die Hilfsalgorithmen cRho-IQ und cUncomputeS zu einem Algorithmus zusammen, der bei der Eingabe eines Ideals das reduzierte Ideal aus der gleichen Äquivalenzklasse berechnet. Dieser Algorithmus ist auf der nächsten Seite abgebildet.

Wir beschreiben kurz die Idee des Algorithmus. Als Eingabe bekommt Reduce-IQ die Diskriminante Δ, eine Zahl $\delta = \lfloor\sqrt{|\Delta|/3}\rfloor$ und das $\mathcal{O}_\Delta$-Ideal (a, b) mit $0 < a \leq |\Delta|/3$ und $0 \leq b < 2a$. Wir führen zuerst den Initialisierungs- und Normalisierungsschritt. Als nächstes wird die Variable state initialisiert, sie wird als Kontrollqubit für die Algorithmen cRho-IQ und cUncomputeS verwendet. Solange $a > \delta$ ist, ist das Ideal noch nicht reduziert und $\mathsf{state} = 1$. Nach dem Satz 2.5.52 erhalten wir das reduzierte Ideal nach höchstens $\log_2(a/\sqrt{|\Delta|}) + 2 \leq (\log \Delta)/2 + 1$ Reduktionsschritten. Wenn a zum ersten Mal kleiner oder gleich δ wird, dann setzen wir $\mathsf{state} = 0$. Ab diesem Zeitpunkt ändern die Algorithmen cRho-IQ und cUncomputeS nichts am vorliegenden Quantenzustand. Nach dem wir cRho-IQ $\lfloor(\log \Delta)/2 + 1\rfloor$ mal wiederholt haben, gilt $a \leq \delta$. Nach dem Satz 2.5.53 müssen wir cRho-IQ höchstens einmal anwenden, damit wir ein reduziertes Ideal bekommen.

Algorithm 24 REDUCE-IQ

Input: $|a\rangle$, $|b\rangle$, Δ, $\delta = \lfloor\sqrt{|\Delta|/3}\rfloor$, wobei $0 < a \leq |\Delta|/3$, $0 < b \leq 2a$ und (a, b) ein $\mathcal{O}_\Delta$-Ideal.
Output: $|a'\rangle$, $|b'\rangle$ und $|\mathsf{temp}\rangle$, wobei (a', b') reduziert ist und temp für die Reversibilität nötige Qubits enthält.

1. Initialisiere $|p\rangle, |\mathsf{counter}\rangle, |\mathsf{state}\rangle$ mit Null und $|q\rangle$ mit Eins. Dabei ist $\text{size}(p) = \text{size}(q) = \log_2|\Delta| + 2$, $\text{size}(\mathsf{counter}) = \lfloor\log\log|\Delta| + 2\rfloor$ und $\text{size}(\mathsf{state}) = 1$.

2. Normalisiere das Ideal (a, b)

$$|b\rangle, |a\rangle, |0\rangle \xrightarrow[\text{CNOT}]{b>a} |b\rangle, |a\rangle, |s_0\rangle$$
$$|s_0\rangle, |a\rangle, |b\rangle \xrightarrow[\text{CSUB}]{s_0=1} |s_0\rangle, |a\rangle, |b - 2s_0a\rangle$$

3. Falls $a > \delta$, setze $\mathsf{state} = 1$

$$|a\rangle, |\mathsf{state}\rangle \xrightarrow[\text{CNOT}]{a>\delta} |a\rangle, |\mathsf{state}_{new}\rangle$$

4. Wiederhole $\lfloor(\log|\Delta|)/2 + 1\rfloor$ mal

   (a) Falls $\mathsf{state} = 1$, berechne neue a, b und s

$$|\mathsf{state}\rangle, |a\rangle, |b\rangle, |0\rangle \xrightarrow{\text{CRHO-IQ}} |\mathsf{state}\rangle, |a_{neu}\rangle, |b_{neu}\rangle, |s\rangle$$

   (b) Falls $\mathsf{state} = 1$, berechne $p' = q$, $q' = p + sq$ und lösche s

$$|\mathsf{state}\rangle, |p\rangle, |q\rangle, |s\rangle, |0\rangle \xrightarrow{\text{CUNCOMPUTES}} |\mathsf{state}\rangle, |p_{new}\rangle, |q_{new}\rangle, |0\rangle, |\operatorname{sgn} p\rangle$$

   (c) Falls zum ersten Mal $a \leq \delta$, setze $\mathsf{state} = 0$

$$|\mathsf{counter}\rangle, |a_{new}\rangle, |\mathsf{state}\rangle \xrightarrow[\text{TOFFOLY}]{\mathsf{counter}=0, a_{new}\leq\delta} |\mathsf{counter}\rangle, |a_{new}\rangle, |\mathsf{state}_{new}\rangle$$

   (d) Falls $\mathsf{state} = 0$, erhöhe $\mathsf{counter}$ um Eins (dieser Schritt ist für die Reversibilität notwendig).

$$|\mathsf{state}\rangle, |\mathsf{counter}\rangle \xrightarrow[\text{CADD}]{\mathsf{state}=0} |\mathsf{state}\rangle, |\mathsf{counter}_{new}\rangle$$

5. Falls (a, b) noch nicht reduziert ist, berechne $\rho(a, b)$.

$$|a\rangle, |b\rangle, ||\Delta|\rangle, |0\rangle \xrightarrow[\text{DIV}]{\text{MULT}} |a\rangle, |b\rangle, |c\rangle, |0\rangle \xrightarrow[3\times\text{TOFFOLY}]{a>c\vee a=c\wedge b<0} |a\rangle, |b\rangle, |c\rangle, |l\rangle$$
$$|a\rangle, |b\rangle, |c\rangle, |l\rangle, |0\rangle \xrightarrow[\text{DIV}]{\text{MULT}} |a\rangle, |b\rangle, ||\Delta|\rangle, |l\rangle, |0\rangle \xrightarrow{\text{CRHO-IQ}} |a_{neu}\rangle, |b_{neu}\rangle, |0\rangle, |l\rangle, |s\rangle$$

6. Gebe aus: $|a\rangle$, $|b\rangle$ und $|p, q, \mathsf{counter}, s_0, s, l, (\text{sgn}(p))_1, (\text{sgn}(p))_2, \ldots, (\text{sgn}(p))_{\lfloor(\log\Delta)/2+1\rfloor}\rangle$

Satz 4.2.4 *Seien Δ eine negative Diskriminante und (a,b) ein $\mathcal{O}_\Delta$-Ideal mit $0 < a \leq |\Delta|/3$ und $0 \leq b < 2a$. Der Algorithmus* REDUCE-IQ *führt die folgende Operation durch:*

$$|a\rangle, |b\rangle, |0\rangle, \Delta, \lfloor\sqrt{|\Delta|/3}\rfloor \longrightarrow |a_{new}\rangle, |b_{new}\rangle, |\mathsf{temp}\rangle, \Delta, \lfloor\sqrt{|\Delta|/3}\rfloor$$

wobei (a_{new}, b_{new}) das reduzierte $\mathcal{O}_\Delta$-Ideal in der Äquivalenzklasse von (a,b) ist.

Für die Ein- und Ausgabe benötigt der Algorithmus $2,5\log_2\Delta + \log_2\log_2\Delta + 12$ Qubits. Außerdem werden höchstens $2,5\log_2\Delta + 10$ Qubits, falls die Quantenaddition verwendet wird, bzw. $3,5\log_2\Delta + 10$ Qubits, falls die klassische Addition verwendet wird, für interne Berechnungen gebraucht.

Sei $n = \mathrm{size}(\Delta)$. Falls klassische Addition verwendet wird, genügen

- $24n^3 + 115n^2 + 114n$ cNOT-*Gatter und*
- $41n^3 + 175n^2 + 64n$ TOFFOLY-*Gatter*

zum Ausführen des Algorithmus. Falls Quantenaddition verwendet wird, sind

- $6n^2 + 23n$ cNOT-*Gatter,*
- $8n^3 + 44n^2 + 49n$ TOFFOLY-*Gatter,*
- $4n^3 + 32n^2 + 78n$ H-*Gatter und*
- $11n^4 + 64n^3 + 176n^2 + 85n$ cR_k-*Gatter*

ausreichend.

Beweis. Wir verwenden die Bezeichnungen aus dem Satz und dem Algorithmus und zeigen zuerst, daß (a_{new}, b_{new}) das reduzierte Ideal in der Äquivalenzklasse von (a,b) ist. Wegen $a \leq |\Delta|/3$ folgt aus dem Satz 2.5.52, daß die Anzahl der Reduktionsschritte höchstens $\log_2(a/\sqrt{|\Delta|})+2 \leq (\log_2|\Delta|)/2+1$ ist. Deshalb ist die Anzahl der Wiederholungen im Schritt 4 ausreichend. Solange $a > \delta$ gilt, wird der Algorithmus cRHO-IQ angewendet und es folgt aus dem Satz 2.5.57, daß $|s| > 1$ ist. Deshalb ist die Vorbedingung von cUNCOMPUTES erfüllt. Wenn $a \leq \delta$ zum ersten Mal auftritt, wird state auf Null gesetzt, so daß ab diesem Moment in der Schleife keine Reduktionsschritte mehr durchgeführt werden. Gilt $a \leq \delta$, dann folgt aus dem Satz 2.5.53, daß nach höchstens einem Schritt das Ideal reduziert ist. Dies wird im Schritt 5 durchgeführt. Die Anzahl der Reduktionsschritte wird durch das Register state kontrolliert.

Als nächstes schätzen wir die Anzahl der Qubits ab, die im Algorithmus verwendet werden. Die Eingabe besteht aus zwei Zahlen a und b mit $\mathrm{size}(a) \leq \lfloor\log_2(|\Delta|/3)\rfloor + 2 \leq \log_2|\Delta| + 1$ und $\mathrm{size}(b) \leq \mathrm{size}(2a) \leq \log_2|\Delta| + 2$. Die Ausgabe besteht aus drei Zahlen: den Zahlen a_{new} und b_{new} mit $-a < b \leq a < \sqrt{|\Delta|/3}$ und der Zahl temp. Für die Größe dieser Zahlen gilt $\mathrm{size}(a_{new}), \mathrm{size}(b_{new}) \leq (\log_2|\Delta|)/2 + 2$ und

$$\begin{aligned}\mathrm{size}(\mathsf{temp}) &\leq \mathrm{size}(p) + \mathrm{size}(\mathsf{counter}) + \mathrm{size}(q) + \mathrm{size}(s_0) + \mathrm{size}(s) + \mathrm{size}(l) + \lfloor(\log|\Delta|)/2 + 1\rfloor \\ &\leq \frac{3}{2}\log_2|\Delta| + \log_2\log_2|\Delta| + 8\end{aligned}$$

Die letzte Ungleichung folgt aus der Tatsache, daß wegen Satz 2.5.54 $p, q \leq \sqrt{|\Delta|}$ und wegen Satz 2.5.58 $|s| \leq 1$ gilt. Zusammen erhalten wir also, daß für die Ein-/Ausgabe

$$2,5 \log_2|\Delta| + \log_2 \log_2|\Delta| + 12$$

Qubits notwendig sind.

Die genaue Analyse ergibt, daß für interne Berechnungen zusätzlich höchstens

$$2,5 \log_2|\Delta| + 10 \quad \text{bzw.} \quad 3,5 \log_2|\Delta| + 10$$

Qubits notwendig sind, wenn die Quantenaddition bzw. die klassische Addition verwendet wird.

Die Anzahl der Quantengatter läßt sich aus den Sätzen 4.2.1, 4.2.2 sowie den Ergebnissen aus dem letzten Kapitel berechnen. □

4.3 Gruppenoperation in der Klassengruppe

In diesem Abschnitt entwickeln wir einen Algorithmus, welcher bei Eingabe von zwei reduzierten Idealen $\mathfrak{a}$ und $\mathfrak{b}$ das reduzierte Ideal $\mathfrak{c}$ in der Äquivalenzklasse von $\mathfrak{a} \cdot \mathfrak{b}$ zurückgibt. Dazu verwenden wir die Algorithmen aus den letzten zwei Abschnitten.

Die Idee des Algorithmus ist die folgende: Als erstes berechnen wir das Ideal $\mathfrak{c}'$, das Produkt der Ideale $\mathfrak{a}$ und $\mathfrak{b}$, und reduzieren es. Dadurch bekommen wir das gesuchte Ideal $\mathfrak{c}$. Während der Reduktionen entstehen zusätzliche Qubits, die für die Reversibilität notwendig sind. Um diese Qubits zu löschen, kopieren wir nun das reduzierte Ideal in die Ausgaberegister und führen die Reduktion und Multiplikation rückwärts aus. Wir erhalten dadurch das Ausgangsideal $\mathfrak{a}$. Als nächstes löschen wir dieses Ideal. Dazu berechnen wir das Produkt des Ideals $\mathfrak{c}$ mit dem Ideal $\mathfrak{b}^{-1}$, reduzieren das Ergebnis und erhalten dadurch das Ideal $\mathfrak{a}$. Nun können wir das Ausgangsideal durch einfaches Kopieren löschen. Schließlich führen wir die letzte Reduktion und Multiplikation wieder rückwärts aus, um das Ideal $\mathfrak{c}$ zu erhalten.

Der Algorithmus ist auf der nächsten Seite abgebildet. Seine Korrektheit, Qubit- und Gatter-Komplexität werden im folgenden Satz untersucht.

Satz 4.3.1 *Sei $\Delta < -16$ eine Diskriminante eines imaginär-quadratischen Zahlkörpers und (a_1, b_1), (a_2, b_2) reduzierte $\mathcal{O}_\Delta$-Ideale. Der Algorithmus* MULTIPLICATION-IQ *führt die folgende Quantenoperation durch:*

$$|\mathsf{ctrl}\rangle, |a_1\rangle, |b_1\rangle, a_2, b_2, \Delta, \lfloor\sqrt{|\Delta|/3}\rfloor \longrightarrow |\mathsf{ctrl}\rangle, |a_{new}\rangle, |b_{new}\rangle, a_2, b_2, \Delta, \lfloor\sqrt{|\Delta|/3}\rfloor,$$

wobei $(a, b) = (a_1, b_1)$ gilt, falls $\mathsf{ctrl} = 0$, und (a, b) das reduzierte Ideal in der Äquivalenzklasse von $(a_1, b_1) \cdot (a_2, b_2)$ ist, falls $\mathsf{ctrl} = 1$ gilt.

Der Algorithmus benötigt für die Ein-/Ausgabe höchstens $\log_2|\Delta| + 5$ Qubits. Zusätzlich werden für die internen Berechnungen höchstens $6,5 \log_2|\Delta| + \log_2 \log_2|\Delta| + 10$ Qubits, falls die Quantenaddition verwendet wird, oder höchstens $7,5 \log_2|\Delta| + \log_2 \log_2|\Delta| + 10$ Qubits, falls die klassische Addition verwendet wird, gebraucht.

Falls klassische Addition verwendet wird, genügen

Algorithm 25 MULTIPLICATION-IQ

Input: Ein Kontrollqubit $|\mathsf{ctrl}\rangle$, eine Diskriminante $\Delta < -16$, $|a_1\rangle$, $|b_1\rangle$, a_2, b_2, wobei (a_1, b_1) und (a_2, b_2) reduzierte $\mathcal{O}_\Delta$-Ideale sind.
Output: $|\mathsf{ctrl}\rangle$, $|a_{out}\rangle$, $|b_{out}\rangle$, wobei (a_{out}, b_{out}) ein reduziertes $\mathcal{O}_\Delta$-Ideal in der Äquivalenzklasse von $(a_1, b_1) \cdot (a_2, b_2)$ ist, falls $\mathsf{ctrl} = 1$, und $(a_{out}, b_{out}) = (a_1, b_1)$ gilt, falls $\mathsf{ctrl} = 0$.

1. Berechne (a, b), das Produkt der Ideale (a_1, b_1) und (a_2, b_2), und reduziere (a, b)

$$|a_1\rangle, |b_1\rangle, |0\rangle, |0\rangle, |0\rangle \xrightarrow{\text{IDEAL-MULT}} |a_1\rangle, |b_1\rangle, |a\rangle, |b\rangle, |0\rangle \xrightarrow{\text{REDUCE-IQ}} |a_1\rangle, |b_1\rangle, |a'\rangle, |b'\rangle, |\mathsf{temp}\rangle$$

2. Falls $\mathsf{ctrl} = 1$, kopiere a' und b' in die Ausgaberegister

$$|\mathsf{control}\rangle, |a'\rangle, |b'\rangle, |0\rangle, |0\rangle \xrightarrow[2\times\text{CCOPY}]{\mathsf{ctrl}=1} |\mathsf{control}\rangle, |a'\rangle, |b'\rangle, |a_{out}\rangle, |b_{out}\rangle$$

3. Falls $\mathsf{ctrl} = 0$, kopiere a_1 und b_1 in die Ausgaberegister

$$|\mathsf{control}\rangle, |a_1\rangle, |b_1\rangle, |0\rangle, |0\rangle \xrightarrow[2\times\text{CCOPY}]{\mathsf{ctrl}=0} |\mathsf{control}\rangle, |a_1\rangle, |b_1\rangle, |a_{out}\rangle, |b_{out}\rangle$$

4. Führe den Schritt 1 rückwärts aus, um temp, a' und b' zu löschen

$$|a_1\rangle, |b_1\rangle, |a'\rangle, |b'\rangle, |\mathsf{temp}\rangle \xrightarrow[\text{IDEAL-MULT}^\dagger]{\text{REDUCE-IQ}^\dagger} |a_1\rangle, |b_1\rangle, |0\rangle, |0\rangle, |0\rangle$$

5. Berechne das Produkt der Ideale (a_{out}, b_{out}) und $(a_2, -b_2)$ und reduziere sie anschließend

$$|a_{out}\rangle, |b_{out}\rangle, |0\rangle, |0\rangle, |0\rangle \xrightarrow[\text{REDUCE-IQ}]{\text{IDEAL-MULT}} |a_{out}\rangle, |b_{out}\rangle, |a''\rangle, |b''\rangle, |\mathsf{temp}\rangle$$

6. Lösche a_1 und b_1 durch das Kopieren (es gilt $a_1 = a''$, $b_1 = b''$, falls $\mathsf{control} = 1$ und $a_1 = a_{out}$, $b_1 = b_{out}$, falls $\mathsf{control} = 0$)

$$|\mathsf{control}\rangle, |a''\rangle, |b''\rangle, |a_1\rangle, |b_1\rangle \xrightarrow[2\times\text{CCOPY}]{\mathsf{ctrl}=1} |\mathsf{control}\rangle, |a''\rangle, |b''\rangle, |a_3\rangle, |b_3\rangle$$

$$|\mathsf{control}\rangle, |a_{out}\rangle, |b_{out}\rangle, |a_3\rangle, |b_3\rangle \xrightarrow[2\times\text{CCOPY}]{\mathsf{ctrl}=0} |\mathsf{control}\rangle, |a_{out}\rangle, |b_{out}\rangle, |0\rangle, |0\rangle$$

7. Führe den Schritt 5 rückwärts aus um, temp, a'' und b'' zu löschen

$$|a_{out}\rangle, |b_{out}\rangle, |a''\rangle, |b''\rangle, |\mathsf{temp}\rangle \xrightarrow[\text{IDEAL-MULT}^\dagger]{\text{REDUCE-IQ}^\dagger} |a_{out}\rangle, |b_{out}\rangle, |0\rangle, |0\rangle, |0\rangle$$

8. Gebe aus: $|\mathsf{ctrl}\rangle$, $|a_{out}\rangle$ und $|b_{out}\rangle$

- $96(\log_2|\Delta|)^3 + 5284(\log_2|\Delta|)^2 + 25128\log_2|\Delta| + 30000$ cNOT-*Gatter und*
- $168(\log_2|\Delta|)^3 + 6616(\log_2|\Delta|)^2 + 29240\log_2|\Delta| + 33360$ Toffoly-*Gatter*

zum Ausführen des Algorithmus. Falls Quantenaddition verwendet wird, sind

- $2040(\log_2|\Delta|)^2 + 8892\log_2|\Delta| + 8160$ cNOT-*Gatter,*
- $1328(\log_2|\Delta|)^3 + 15608(\log_2|\Delta|)^2 + 58820\log_2|\Delta| + 65832$ Toffoly-*Gatter,*
- $16(\log_2|\Delta|)^3 + 2624(\log_2|\Delta|)^2 + 13976\log_2|\Delta| + 17584$ H-*Gatter und*
- $44(\log_2|\Delta|)^4 + 3056(\log_2|\Delta|)^3 + 23812(\log_2|\Delta|)^2 + 63592\log_2|\Delta| + 56000$ cR_k-*Gatter*

ausreichend. □

4.4 Berechnung der Ordnung eines Elements

Wir präsentieren nun einen Algorithmus zur Berechnung der Ordnung eines Gruppenelements in der Klassengruppe eines imaginär-quadratischen Zahlkörpers. Dazu definieren wir zuerst eine periodische Funktion, deren Periode die gesuchte Ordnung ist. Die Periode berechnen wir mit dem Quanten-Framework aus dem letzten Kapitel.

Satz 4.4.1 *Seien Δ eine Diskriminante eines imaginär-quadratischen Zahlkörpers und $\mathfrak{a}$ ein $\mathcal{O}_\Delta$-Ideal. Die Funktion* Ord_{IQ} *sei wie folgt definiert* $\mathrm{Ord}_{IQ} : \mathbb{N} \longrightarrow \mathcal{R}_\Delta : x \longmapsto \mathfrak{a}^x$.

Ist m die Ordnung von $\mathfrak{a}$ in der Klassengruppe Cl_Δ, dann ist $m\mathbb{Z}$ das Periodengitter von Ord_{IQ}. Das zu $m\mathbb{Z}$ duale Gitter ist $1/m\mathbb{Z}$.

Algorithmus SampleDual-Ord-IQ, welcher auf der nächsten Seite abgebildet ist, berechnet Approximationen von Vektoren des Gitters $1/m\mathbb{Z}$. Seine Korrektheit und Qubit-Komplexität werden im folgenden Satz untersucht.

Satz 4.4.2 *Seien Δ eine Diskriminante eines imaginär-quadratischen Zahlkörpers, $\mathfrak{a}$ ein $\mathcal{O}_\Delta$-Ideal, m die Ordnung von $\mathfrak{a}$ in Cl_Δ und* Ord_{IQ} *eine Funktion aus der Definition 4.4.1. Bei Eingabe von Δ, einer Zweierpotenz q mit $q \geq (\Delta \ln^2 \Delta)/4$ und eines reduzierten $\mathcal{O}_\Delta$-Idealen $\mathfrak{a}_i$ mit $\mathfrak{a}_i \sim \mathfrak{a}^i$ für alle $i = 1, \ldots, \log_2 q$, berechnet der Algorithmus* SampleDual-Ord-IQ *eine Approximation y eines zufälligen Vektors z aus dem Gitter $(1/m)\mathbb{Z}$, welches zum Periodengitter von* Ord_{IQ} *dual ist. Dabei gilt $|y - z| \leq 1/(4q)$.*

Der Algorithmus benötigt für die internen Berechnungen höchstens $7,5\log_2|\Delta| + \log_2\log_2\Delta + 16$ *Qubits, falls die Quantenaddition verwendet wird, oder höchstens* $8,5\log_2\Delta + \log_2\log_2\Delta + 16$ *Qubits, falls die klassische Addition verwendet wird.*

Die Erfolgswahrscheinlichkeit des Algorithmus ist mindestens $(m-1)/(4m)$.

Algorithm 26 SampleDual-Ord-IQ

Input: $q \in \mathbb{N}$, eine negative Diskriminante Δ, eine reduziertes $\mathcal{O}_\Delta$-Ideal $\mathfrak{a}$ und reduzierte $\mathcal{O}_\Delta$-Ideale $\mathfrak{a}_1, \ldots, \mathfrak{a}_{\log q}$ mit $\mathfrak{a}_i \sim \mathfrak{a}^i$, $1 \leq i \leq \log q$.
Output: Eine Zahl y, wobei y/q eine "gute" Approximation eines Vektor z/m aus dem Gitter $(1/m)\mathbb{Z}$ ist, d.h. es gilt $y/q = z/m + \omega$ mit $|\omega| \leq 1/(4q)$.

1. Initialzustand
$$|0\rangle, |(1, \Delta \mod 2)\rangle.$$

2. Erzeuge Superposition
$$\overset{(\log q)\times \mathrm{H}}{\longrightarrow} \frac{1}{\sqrt{q}} \sum_{x=0}^{q-1} |x\rangle, |(1, \Delta \mod 2)\rangle.$$

3. Berechne die Funktion Ord_{IQ}. Dazu wende $(\log_2 q)$ mal die Funktion Multiplication-IQ an. Für $1 \leq i \leq \log_2 q$ ist q_i (das i-te Bit von q) das Steuerungsqubit, $\mathfrak{a}_i$ ist klassisch gegeben und $|a\rangle$ und $|b\rangle$ stehen im dem zweiten Register.
$$\overset{(\log q)\times \text{Multiplication-IQ}}{\longrightarrow} \frac{1}{\sqrt{q}} \sum_{x=0}^{q-1} |x\rangle, |\operatorname{Ord}_{IQ}(x)\rangle.$$

4. Messe das zweite Register
$$\longrightarrow \frac{1}{\sqrt{p}} \sum_{x=0}^{p-1} |x' + xm\rangle |\operatorname{Ord}_{IQ}(x')\rangle$$
wobei x' ein zufälliges Element aus der Menge $\{0, \ldots, m-1\}$ und $p = \left\lfloor \frac{q-x'-1}{m} \right\rfloor + 1$ gilt.

5. Wende die Quantenfouriertransformation auf das erste Register an
$$\overset{\mathrm{QFT}}{\longrightarrow} \frac{1}{\sqrt{2qp}} \sum_{y=0}^{2q-1} \sum_{x=0}^{p-1} \exp(2\pi i \frac{(x' + xm)y}{2q}) |y\rangle |\operatorname{Ord}_{IQ}(x', \mathfrak{a})\rangle.$$

6. Messe das erste Register und gebe $y/2q$ aus.

Beweis. Als erstes analysieren wir die Erfolgswahrscheinlichkeit des Algorithmus, d.h. die Wahrscheinlichkeit daß der Algorithmus eine "gute" Approximation eines Vektors aus $1/m\mathbb{Z}$ zurückgibt. Wir nennen die Approximation gut, wenn die Zahl y, die vom Algorithmus zurückgegeben wird, in der folgenden Menge liegt:

$$\mathcal{Y} = \{\, y \in \mathbb{N} \mid 0 \leq y < 2q, \text{ und } \frac{y}{2q} = \frac{z}{m} + \omega \text{ mit } |\omega| < \frac{1}{4q} \,\}. \tag{4.2}$$

Die Wahrscheinlichkeit eine bestimmte Zahl $y \in \mathcal{Y}$ zu messen ist

$$\frac{1}{2pq}\left|\sum_{x=0}^{p-1}\exp(2\pi i\frac{(x'+xm)y}{2q})\right|^2 = \frac{1}{2pq}\left|\sum_{x=0}^{p-1}\exp(2\pi i\frac{xmy}{2q})\right|^2,$$

wobei $p = \left\lfloor \frac{q-x'-1}{m} \right\rfloor + 1$ und $x' \in \{0,\ldots,m-1\}$. Wegen

$$xmy/(2q) \equiv xm(z/m+\omega) \equiv xm\omega \mod 1$$

mit z und ω aus (4.2) gilt

$$\frac{1}{2pq}\left|\sum_{x=0}^{p-1}\exp(2\pi i\frac{xmy}{2q})\right|^2 = \frac{1}{2pq}\left|\sum_{x=0}^{p-1}\exp(2\pi ixm\omega)\right|^2 \geq \frac{1}{2pq}\left(p\frac{\sqrt{2}}{2}\right)^2 = \frac{p}{4q} \geq \frac{1}{4m},$$

wobei die erste Ungleichung deshalb gilt, weil p Vektoren der Länge Eins aus einem Kreissegment mit dem Winkel höchstens $\pi/2$ addiert werden.

Als nächstes schätzen wir $\operatorname{card}\mathcal{Y}$ von unten ab. Es gilt

$$\begin{aligned}\operatorname{card}\mathcal{Y} &= \operatorname{card}\{\, x \in \mathbb{N} \mid 0 \leq y < 2q, \text{ und } \frac{y}{2q} = \frac{z}{m} + \omega \text{ mit } |\omega| < \frac{1}{4q} \,\} = \\ &\operatorname{card}\{\, z \in \mathbb{N} \mid 0 \leq \frac{z}{m} + \omega < 1 \,\} \geq \operatorname{card}\{\, z \in \mathbb{N} \mid \frac{1}{4q} \leq \frac{z}{m} \leq 1 - \frac{4}{q} \,\} \geq m-1.\end{aligned}$$

Die letzte Ungleichung folgt aus der Tatsache, daß $m \leq \sqrt{q}$ ist.

Wir fassen die beiden letzten Ergebnisse zusammen und erhalten die untere Schranke für die Wahrscheinlichkeit, daß SampleDual-Ord-IQ eine "gute" Approximation eines Vektors aus dem Gitter $1/m\mathbb{Z}$ liefert, nämlich

$$\frac{m-1}{4m}.$$

Als nächstes analysieren wir die Anzahl der Qubits. Da wir die semiklassische Quantenfouriertransformation verwenden, brauchen wir nur noch ein Qubit zum Speichern von x im ersten Register. Außerdem sind $\log\Delta + 4$ Qubits für das zweite Register notwendig, da darin ein $\mathcal{O}_\Delta$-Ideal gespeichert werden muß. Intern ruft SampleDual-Ord-IQ den Algorithmus Multiplication-IQ $\log_2 q$ mal auf. Deshalb folgt aus dem Satz 4.3.1, daß die Anzahl der Qubits höchstens $7,5\log_2\Delta + \log_2\log_2\Delta + 16$ bzw. $8,5\log_2\Delta + \log_2\log_2\Delta + 16$ ist, falls die Quantenaddition bzw. die klassische Addition verwendet wird. □

Als nächstes präsentieren den Algorithmus zur Berechnung der Ordnung eines Ideals in der Klassengruppe eines imaginär-quadratischen Zahlkörpers.

Algorithm 27 ORD-IQ

Input: Eine negative Diskriminante Δ, ein $\mathcal{O}_\Delta$-Ideal $\mathfrak{a}$.
Output: Die Ordnung m von $\mathfrak{a}$ in Cl_Δ.

1. Setze $q = 2^x$, so daß $q/2 < (|\Delta| \ln^2 |\Delta|)/9 \leq q$.
2. Berechne klassisch $\mathfrak{a}_i = \mathfrak{a}^i$ für $i = 1, \ldots, \log_2 q$.
3. Berechne mit SAMPLEDUAL-ORD-IQ zwei Vektoren $y_1/(2q)$ und $y_2/(2q)$, welche zwei Vektoren aus $1/m\mathbb{Z}$ approximieren.
4. Mit dem Algorithmus zur Kettenbruchentwicklung berechne $s_1,\ m_1$ und $s_2,\ m_2$, so daß
$$\left| \frac{s_i}{m_i} - \frac{y_i}{2q} \right| \leq \frac{1}{4q} \quad \text{für } i = 1, 2$$
gilt.
5. Gebe das kleinste gemeinsame Vielfache von m_1 und m_2 aus.

Satz 4.4.3 *Seien $\Delta \leq -16$ eine Diskriminante eines imaginär-quadratischen Zahlkörpers, $\mathfrak{a}$ ein $\mathcal{O}_\Delta$-Ideal und Ord_{IQ} eine Funktion aus der Definition 4.4.1. Bei Eingabe von Δ und $\mathfrak{a}$ berechnet der Algorithmus* ORD-IQ *die Ordnung von $\mathfrak{a}$ in Cl_Δ.*

Der Algorithmus benötigt für die internen Berechnungen höchstens $7,5 \log_2 \Delta + \log_2 \log_2 \Delta + 16$ *Qubits, falls die Quantenaddition verwendet wird, oder höchstens* $8,5 \log_2 \Delta + \log_2 \log_2 \Delta + 16$ *Qubits, falls die klassische Addition verwendet wird.*

Falls die klassische Addition verwendet wird, genügen

- $4 \log_2|\Delta| + 8 \log_2 \log_2|\Delta|$ H-*Gatter,*
- $4(\log_2|\Delta|)^2$ cR_k-*Gatter,*
- $384(\log_2|\Delta|)^4 + 21136(\log_2|\Delta|)^3 + 100512(\log_2|\Delta|)^2 + 120000 \log_2|\Delta|$ CNOT-*Gatter und*
- $672(\log_2|\Delta|)^4 + 26464(\log_2|\Delta|)^3 + 116960(\log_2|\Delta|)^2 + 133440 \log_2|\Delta|$ TOFFOLY-*Gatter*

zum Ausführen des Algorithmus. Falls die Quantenaddition verwendet wird, sind

- $8160(\log_2|\Delta|)^3 + 35568(\log_2|\Delta|)^2 + 32640 \log_2|\Delta|$ CNOT-*Gatter,*
- $5312(\log_2|\Delta|)^4 + 62432(\log_2|\Delta|)^3 + 235280(\log_2|\Delta|)^2 + 263328 \log_2|\Delta|$ TOFFOLY-*Gatter,*
- $64(\log_2|\Delta|)^4 + 10496(\log_2|\Delta|)^3 + 55904(\log_2|\Delta|)^2 + 70344 \log_2|\Delta|$ H-*Gatter und*
- $176(\log_2|\Delta|)^5 + 12224(\log_2|\Delta|)^4 + 95248(\log_2|\Delta|)^3 + 254372(\log_2|\Delta|)^2 + 224000 \log_2|\Delta|$ cR_k-*Gatter*

ausreichend.

Die Erfolgswahrscheinlichkeit des Algorithmus ist mindestens $1/256$.

Beweis. Wir analysieren zuerst die Erfolgswahrscheinlichkeit von ORD-IQ. Sie setzt sich zusammen aus der Wahrscheinlichkeit, daß im Schritt 3 zwei "gute" Approximationen von Zahlen in $1/m\mathbb{Z}$ berechnet werden, und der Wahrscheinlichkeit, daß die Zahlen s_1 und s_2, die im Schritt 4 berechnet werden, teilerfremd sind. Wegen der Sätzen 4.4.2 und 2.2.3 ist die untere Schranke für die Erfolgswahrscheinlichkeit von ORD-IQ mindestens 1/256.

Die Wahl von q im Schritt 1 garantiert uns, daß die Bedingung im Satz 2.3.1 erfüllt ist und deshalb der Algorithmus im Schritt 4 einen Teiler von m berechnet.

Die Anzahl der benötigten Qubits ist die gleiche wie im Algorithmus SAMPLEDUAL-ORD-IQ. Die Anzahl der elementaren Quantengatter folgt aus dem Satz 4.3.1 (MULTIPLICATION-IQ wird höchstens $(|\Delta| \ln^2|\Delta|)/9$ aufgerufen) und den Ergebnissen aus dem letzten Kapitel. □

4.5 Berechnung des diskreten Logarithmus

Seien $\mathfrak{a}$ und $\mathfrak{b}$ $\mathcal{O}_\Delta$-Ideale. Und es gelte $\mathfrak{a}^n = \mathfrak{b}$, wobei n minimal gewählt ist. In diesem Abschnitt präsentieren wir einen Algorithmus, welcher bei Eingabe von $\mathfrak{a}$ und $\mathfrak{b}$ die Zahl n, den diskreten Logarithmus von $\mathfrak{b}$ zur Basis $\mathfrak{a}$, berechnet.

Satz 4.5.1 *Seien $\mathfrak{a}$ und $\mathfrak{b}$ fix und $m = \text{order}(\mathfrak{a})$ in Cl_Δ. Setze $q = 2^x$, $x \in \mathbb{N}$, so daß $q/2 < m \leq q$. Die Funktion DL_{IQ} sei wie folgt definiert*

$$\begin{aligned} \text{DL}_{IQ} : \{0, \ldots, q-1\}^2 &\longrightarrow \mathcal{R}_\Delta \\ (x, y) &\longmapsto \mathfrak{a}^x \mathfrak{b}^y. \end{aligned}$$

Die Funktion DL_{IQ} kann mit schneller Exponentiation mit Hilfe von MULTIPLICATION-IQ berechnet werden. Dabei können die reduzierten Ideale in den Äquivalenzklassen von $\mathfrak{a}$, $\mathfrak{a}^2$, $\mathfrak{a}^4$, ..., $\mathfrak{a}^{q/2}$, $\mathfrak{b}$, $\mathfrak{b}^2$, $\mathfrak{b}^4$, ..., $\mathfrak{b}^{q/2}$ klassisch bestimmt werden.

Eine Basis des Periodengitters L von DL_{IQ} besteht aus den Spaltenvektoren der Matirx

$$\begin{pmatrix} m & m-n \\ 0 & 1 \end{pmatrix}.$$

Eine Basis des dualen Gitters L^* ist besteht aus den Spaltenvektoren der Matrix

$$\begin{pmatrix} 1/m & 0 \\ n/m & 1 \end{pmatrix}.$$

Satz 4.5.2 *Algorithmus* DL-IQ*, welcher auf der folgenden Seite angegeben ist, berechnet den diskreten Logarithmus in der Klassengruppe der Ideale eines imaginär-quadratischen Zahlkörpers der Diskriminante Δ.*

Der Algorithmus benötigt für die internen Berechnungen höchstens $7,5 \log_2 \Delta + \log_2 \log_2 \Delta +$ 16 Qubits, falls die Quantenaddition verwendet wird, oder höchstens $8,5 \log_2 \Delta + \log_2 \log_2 \Delta +$ 16 Qubits, falls die klassische Addition verwendet wird.

Falls die klassische Addition verwendet wird, genügen

Algorithm 28 DL-IQ

Input: Eine negative Diskriminante Δ und $\mathcal{O}_\Delta$-Ideale $\mathfrak{a}$, $\mathfrak{b}$.
Output: Der diskrete Logarithmus von $\mathfrak{b}$ zur Basis $\mathfrak{a}$.

1. Mit ORD-IQ berechne m, die Ordnung von $\mathfrak{a}$ in Cl_Δ.
2. Initialzustand
$$|0\rangle, |0\rangle, |(1, \Delta \mod 2)\rangle.$$
3. Setze $q = 2^x$, $x \in \mathbb{N}$, so daß $q/2 < m \leq q$.
4. Erzeuge Superposition
$$\xrightarrow{(2\log q)\times \mathrm{H}} \frac{1}{q} \sum_{x_1=0}^{q-1} \sum_{x_2=0}^{q-1} |x_1\rangle, |x_2\rangle, |(1, \Delta \mod 2)\rangle.$$
5. Berechne die Funktion DL_{IQ}
$$\xrightarrow{(2\log q)\times \text{MULTIPLICATION-IQ}} \frac{1}{q} \sum_{x_1=0}^{q-1} \sum_{x_2=0}^{q-1} |x_1\rangle, |x_2\rangle, |\,\mathrm{DL}_{IQ}(x_1, x_2)\rangle.$$
6. Messe das letzte Register
$$\frac{1}{\sqrt{p}} \sum_{\mathbf{x}\in\mathcal{M}} |\mathbf{x}\rangle |\,\mathrm{DL}_{IQ}(x_1', x_2')\rangle$$
mit $\mathcal{M} = \{\,(x_1, x_2) \in \mathbb{Z}^2 \mid \mathrm{DL}_{IQ}(x_1, x_2) = \mathrm{DL}_{IQ}(x_1', x_2') \text{ und } 0 \leq x_1, x_2 < q\,\}$ und $p = \operatorname{card} \mathcal{M}$.
7. Wende die Quantenfouriertransformation auf die ersten zwei Register an
$$\xrightarrow{\mathrm{QFT}} \frac{1}{4q\sqrt{p}} \sum_{y_1,y_2=0}^{4q-1} \sum_{\mathbf{x}\in\mathcal{M}} \exp(2\pi i \frac{\mathbf{x}\cdot\mathbf{y}}{4q}) |y_1\rangle |y_2\rangle |\,\mathrm{DL}_{IQ}(x_1', x_2')\rangle.$$
8. Messe die ersten zwei Register. Seien y_1 und y_2 die gemessenen Zahlen.
9. Berechne $z_1 = \lfloor y_1 m/(4q) \rceil$.
10. Mit dem erweiterten euklidischen Algorithmus berechne t_1 und t_2, so daß $t_1 z_1 + t_2 m = \gcd(z_1, m)$.
11. Gebe aus: $t_1 \lfloor m y_2/(4q) \rceil / \gcd(z_1, m) \mod m$.

- $8\log_2|\Delta| + 10\log_2\log_2|\Delta|$ H-*Gatter,*
- $5(\log_2|\Delta|)^2$ cR_k-*Gatter,*
- $576(\log_2|\Delta|)^4 + 31704(\log_2|\Delta|)^3 + 150768(\log_2|\Delta|)^2 + 180000\log_2|\Delta|$ cNOT-*Gatter und*
- $1008(\log_2|\Delta|)^4 + 39696(\log_2|\Delta|)^3 + 175440(\log_2|\Delta|)^2 + 200160\log_2|\Delta|$ Toffoly-*Gatter*

zum Ausführen des Algorithmus. Falls die Quantenaddition verwendet wird, sind

- $12240(\log_2|\Delta|)^3 + 53352(\log_2|\Delta|)^2 + 48960\log_2|\Delta|$ cNOT-*Gatter,*
- $7968(\log_2|\Delta|)^4 + 93648(\log_2|\Delta|)^3 + 352920(\log_2|\Delta|)^2 + 394992\log_2|\Delta|$ Toffoly-*Gatter,*
- $96(\log_2|\Delta|)^4 + 15744(\log_2|\Delta|)^3 + 83856(\log_2|\Delta|)^2 + 105517\log_2|\Delta|$ H-*Gatter und*
- $264(\log_2|\Delta|)^5 + 18336(\log_2|\Delta|)^4 + 142872(\log_2|\Delta|)^3 + 381556(\log_2|\Delta|)^2 + 336000\log_2|\Delta|$ cR_k-*Gatter*

ausreichend.

Die Erfolgswahrscheinlichkeit ist mindestens $1/2^{15}$*.*

Beweis. Wir verwenden die gleichen Bezeichnungen wie im Algorithmus DL-IQ. Darüberhinaus setzen wir

$$\mathcal{Y} = \{\,(y_1,y_2) \in \mathbb{Z}^2 \mid 0 \le y_1, y_2 < 4q,\ \frac{y_1}{4q} = \frac{z_1}{m} + \omega_1,\ \frac{y_2}{4q} = \frac{n}{m}z_1 + z_2 + \omega_2, \tag{4.3}$$

$$\text{mit } z_1, z_2 \in \mathbb{Z} \text{ und } |\omega_1|, |\omega_2| \le \frac{1}{8q}\,\}. \tag{4.4}$$

Wir beweisen als erstes die Korrektheit des Algorithmus unter der Annahme, daß im Schritt 8 des Algorithmus der Vektor $(y_1,y_2) \in \mathcal{Y}\backslash\{(0,0)\}$ gemessen wurde. Die Wahrscheinlichkeit, mit der solche Vektoren gemessen werden, schätzen wir weiter unten ab.

Aus $(y_1,y_2) \in \mathcal{Y}$ folgt, daß zwei ganze Zahlen z_1 und z_2 existieren, so daß

$$\frac{y_1}{4q} = \frac{z_1}{m} + \omega_1,\ \text{mit } |\omega_1| \le \frac{1}{8q} \quad \text{und} \quad \frac{y_2}{4q} = \frac{n}{m}z_1 + z_2 + \omega_1,\ \text{mit } |\omega_1| \le \frac{1}{8q}. \tag{4.5}$$

Aus (4.5) folgt, daß

$$z_1 = \left(\frac{y_1}{4q} - \omega_1\right) m.$$

Aus der Tatsache, daß z_1 eine ganze Zahl ist und $|\omega_1|m \le m/8q < 1/2$ folgt

$$z_1 = \left\lfloor \frac{y_1 m}{4q} \right\rceil.$$

Außerdem folgt aus (4.5), daß

$$\frac{m y_2}{4q} - m\omega_2 = n z_1 + m z_2$$

gilt. Auch in diesem Fall ist $|\omega_2|m < 1/2$ und $nz_1 + mz_2 \in \mathbb{Z}$, so daß die Gleichung

$$nz_1 + mz_2 = \left\lfloor \frac{my_2}{4q} \right\rceil \tag{4.6}$$

erfüllt ist. Der erweiterte euklidische Algorithmus im Schritt 10 berechnet zwei Zahlen t_1 und t_2 mit $t_1 z_1 + t_2 m = \gcd(z_1, m)$. Zusammen mit (4.6) ergibt das

$$\left\lfloor \frac{my_2}{4q} \right\rceil = \frac{t_1 z_1}{\gcd(z_1,m)} \left\lfloor \frac{my_2}{4q} \right\rceil + \frac{t_2 m}{\gcd(z_1,m)} \left\lfloor \frac{my_2}{4q} \right\rceil = rz_1 + m\left(q + \frac{t_2}{\gcd(z_1,m)} \left\lfloor \frac{my_2}{4q} \right\rceil\right)$$

mit $r + mq = t_1 \lfloor my_2/(4q) \rceil / \gcd(z_1, m)$. Wir wissen, daß $0 \le n < m$. Deshalb ist $n = r$ der gesuchte diskrete Logarithmus.

Als nächstes schätzen wir die Erfolgswahrscheinlichkeit von DL-IQ von unten ab, d.h. die Wahrscheinlichkeit, daß im Schritt 1 die Zahl m und im Schritt 8 das Paar $(y_1, y_2) \in \mathcal{Y}$ gemessen werden.

Nach Satz 4.4.3 ist die Wahrscheinlichkeit, daß im ersten Schritt m gemessen wird mindestens $1/256$. Die Wahrscheinlichkeit, daß ein bestimmtes Element $(y_1, y_2) \in \mathcal{Y}$ zu messen, ist

$$\frac{1}{16q^2p} \left| \sum_{\mathbf{x} \in \mathcal{M}} \exp(2\pi i \frac{\mathbf{x} \cdot \mathbf{y}}{4q}) \right|^2 = \frac{1}{16q^2p} \left| \sum_{\mathbf{x} \in \mathcal{M}'} \exp(2\pi i \frac{\mathbf{x} \cdot \mathbf{y}}{4q}) \right|^2,$$

mit $\mathcal{M}' = \{\, (x_1, x_2) \in L \mid 0 \le x_1' + x_1, x_2' + x_2 < q \,\}$.

Für $\mathbf{x} \in \mathcal{M}'$ und $\mathbf{y} \in \mathcal{Y}$ gilt

$$\frac{\mathbf{x} \cdot \mathbf{y}}{4q} \equiv \mathbf{x}\omega \mod 1.$$

Deshalb folgt aus der Periodizität von exp

$$\frac{1}{16q^2p} \left| \sum_{\mathbf{x} \in \mathcal{M}'} \exp(2\pi i \frac{\mathbf{x} \cdot \mathbf{y}}{4q}) \right|^2 = \frac{1}{16q^2p} \left| \sum_{\mathbf{x} \in \mathcal{M}'} \exp(2\pi i (x_1\omega_1 + x_2\omega_2)) \right|^2 \ge$$
$$\frac{1}{16q^2p} \left| p\frac{\sqrt{2}}{2} \right|^2 = \frac{p}{32q^2}$$

Als nächstes schätzen wir $p = \operatorname{card} \mathcal{M} = \operatorname{card} \mathcal{M}'$ und $\operatorname{card} \mathcal{Y}$ von unten ab. Es gilt

$$\begin{aligned} p = \operatorname{card} \mathcal{M}' = \operatorname{card} \{\, (x_1, x_2) \in L \mid & -x_1' \le x_1 < q - x_1', \text{ und } -x_2' \le x_2 < q - x_2' \,\} = \\ \operatorname{card} \{\, (x_1, x_2) \in L \mid & -x_1' \le mt_1 + (m-n)t_2 < q - x_1', \text{ und} \\ & -x_2' \le t_2 < q - x_2', \ t_1, t_2 \in \mathbb{Z} \,\}, \end{aligned}$$

woraus folgt $p \ge q^2/(2m)$. Darüberhinaus gilt

$$\begin{aligned} \operatorname{card} \mathcal{Y} = \operatorname{card} \{\, (z_1, z_2) \in \mathbb{Z} \mid 0 \le \frac{z_1}{m} + \omega_1 < 1 \text{ und } 0 \le \frac{n}{m} z_1 + z_2 + \omega_2 \text{ mit } |\omega_1|, |\omega_2| \le \frac{1}{8q} \,\} \\ \operatorname{card} \{\, (z_1, z_2) \in \mathbb{Z} \mid -\omega_1 \le \frac{z_1}{m} < 1 - \omega_1 \text{ und} \\ -\omega_2 - \frac{n}{m} z_1 \le z_2 < 1 - \omega_2 - \frac{n}{m} z_1, |\omega_1|, |\omega_2| \le \frac{1}{8q} \,\} \ge m - 1. \end{aligned}$$

Wir fassen alles zusammen und bekommen als untere Schranke für die Erfolgswahrscheinlichkeit von DL-IQ:

$$\frac{1}{256}\frac{p}{32q^2}\,\mathrm{card}(\mathcal{Y}\backslash\{(0,0)\}) \geq \frac{1}{2^{15}}, \quad \text{für } m > 1$$

Die Anzahl der benötigten Qubits ist die gleiche wie im Algorithmus SampleDual-Ord-IQ.

Die Anzahl der elementaren Quantengatter folgt sofort aus den Satz 4.4.3 und den Ergebnissen aus dem letzten Kapitel. □

Kapitel 5

Algorithmen für reell-quadratische Zahlkörper

In diesem Kapitel werden Algorithmen zur Berechnung des Regulators, der Ordnung eines Ideals, des erweiterten diskreten Logarithmus und zur Lösung des Hauptidealproblems in reell-quadratischen Zahlkörpern präsentiert und untersucht.

Im Vergleich zum imaginär-quadratischen Fall ist der reell-quadratische Fall algorithmisch komplizierter, weil die Idealklassen reell-quadratischer Zahlkörper keine eindeutigen Repräsentanten enthalten, die in Polynomzeit berechnet werden können. Darüberhinaus muß in den Algorithmen der natürliche Logarithmus berechnet werden. Diese Berechnung kann jedoch nicht exakt durchgeführt werden, wodurch zusätzliche algorithmische Schwierigkeiten entstehen.

Das Kapitel ist wie folgt aufgebaut: Im ersten Abschnitt werden Algorithmen zum Rechnen mit Idealen reell-quadratischer Zahlkörper vorgestellt. Im zweiten Abschnitt wird ein Algorithmus zur Berechnung des Regulators präsentiert. Dazu wird eine periodische Funktion entworfen, welche nicht injektiv innerhalb einer Periode ist. Auf diese Funktion wird das Quantenframework aus Kapitel 3 angewendet. Es wird gezeigt, daß damit auch die Periode von bestimmten nicht injektiven Funktionen berechnet werden kann. Im nächten Abschnitt wird der Regulator-Algorithmus zu Algorithmen erweitert, welche das Hauptidealproblem, das Ordnungsproblem und das erweiterte DL-Problem lösen. Zu allen Algorithmen wird eine präzise obere Schranke für die Anzahl der Qubits, die für die Ausführung der Algorithmen notwendig sind, angegeben. Für die Anzahl der elementaren Quantengatter werden asymptotische obere Schranken bestimmt.

Im weiteren nehmen wir an, daß $\Delta > 2^{64}$ gilt. Für die Lösung der hier vorgestellten Probleme in Zahlkörpern, dessen Diskriminante kleiner als oder gleich 2^{64} ist, können klassische Algorithmen und Computer verwendet werden.

5.1 Reduktion von Idealen

In diesem Abschnitt entwickeln wir einen Algorithmus zur Reduktion von reell-quadratischen Idealen. Als nächstes zeigen wir im Algorithmus cRho-RQ (siehe Seite 81), wie man einen

Reduktionsschritt durchführt. Danach wird im Algorithmus cDistance (siehe Seite 82) die Berechnung des Abstands zwischen einem Ideal $\mathfrak{a}$ und $\rho(\mathfrak{a})$ beschrieben. Um Qubits zu sparen, wird hier der doppelte Abstand, wie er in der Definition 2.5.68 definiert wird, verwendet. Dadurch wird die Division durch zwei bei Logarithmenberechung überflüssig. Die notwendigen Eigenschaften des Abstands bleiben jedoch erhalten. Schließlich präsentieren wir den Reduktionsalgorithmus Reduce-RQ (siehe Seite 83).

In den folgenden Sätzen untersuchen wir die Laufzeit und den Speicherbedarf der Algorithmen.

Satz 5.1.1 *Sei Δ eine Diskriminante eines reell-quadratischen Zahlkörpers und (a,b) ein $\mathcal{O}_\Delta$-Ideal mit $|a| \leq \Delta$, $|b| \leq 2|a|$. Der Algorithmus* cRho-RQ *führt die folgende Quantenoperation durch:*

$$|\mathsf{ctrl}\rangle, |a\rangle, |b\rangle, |0\rangle, \Delta \longrightarrow |\mathsf{ctrl}\rangle, |a'\rangle, |b'\rangle, |s\rangle,$$

wobei $(a',b') = \rho(a,b)$ und $s = \operatorname{sign}(a')\lfloor b+q)/(2|a'|)\rfloor$, falls $\mathsf{ctrl} = 1$ gilt, und $(a',b') = (a,b)$ und $s = 0$, falls $\mathsf{ctrl} = 0$ gilt. Dabei gilt $q = \lfloor\sqrt{\Delta}\rfloor$, falls $|a'| \leq \lfloor\sqrt{\Delta}\rfloor$, und $q = |a'|$, falls $|a'| > \lfloor\sqrt{\Delta}\rfloor$.

Sei $n = \operatorname{size}(\Delta)$. Der Algorithmus benötigt für die Ein-/Ausgabe höchstens $3n+3$ Qubits. Zusätzlich werden für die internen Berechnungen höchstens $n+4$ Qubits, falls die Quantenaddition verwendet wird, und höchstens $2n+4$ Qubits, falls die klassische Addition verwendet wird, benötigt.

Die Anzahl der elementaren Quantengatter ist $O(n^2)$, falls die klassische Addition verwendet wird, bzw. $O(n^3)$, falls die Quantenaddition verwendet wird.

Beweis. Es ist leicht einzusehen, daß cRho-RQ einen Reduktionsschritt durchführt, wie er in der Definition 2.5.50 beschrieben ist.

Wir bestimmen die Anzahl der Qubits. Für die Ein-/Ausgabe werden $\operatorname{size}(a) + \operatorname{size}(b) + \operatorname{size}(s) + 1$ Qubits benötigt. Nach Voraussetzung gilt $|b| \leq |2a| \leq 2\Delta$. Seien s_i, p_{i-1}, p_i p_{i+1} wie im Satz 2.5.54 definiert, dann folgt $s \leq s_i p_i \leq p_{i+1} + p_{i-1} \leq \Delta + 3\sqrt{\Delta} \leq 2\Delta$ (für $\Delta > 16$). Daraus folgt, daß für die Ein-/Ausgabe höchstens $3n+3$ Qubits gebraucht werden. Die Anzahl der Qubits, die für die interne Berechnungen verwendet werden, sowie die Anzahl der elementaren Gatter können mit den Ergebnissen aus dem Kapitel 3 einfach berechnet werden. □

Satz 5.1.2 *Seien Δ eine Diskriminante eines reell-quadratischen Zahlkörpers und (a,b) ein $\mathcal{O}_\Delta$-Ideal mit $|a| \leq \Delta$, $|b| \leq 2|a|$. Außerdem seien $p,k \in \mathbb{N}$ mit $p > \max\{64, \log\Delta\}$ und $d \in 2^{-p}\mathbb{Z}$ mit $\lfloor|d|\rfloor \leq 2^k < 2^5\Delta$. Der Algorithmus* cDistance *führt die folgende Quantenoperation durch:*

$$|\mathsf{ctrl}\rangle, |b\rangle, |d\rangle, \Delta, \lceil\sqrt{\Delta}\rfloor_p, p, k \longrightarrow |\mathsf{ctrl}\rangle, |b\rangle, |d + \mathsf{ctrl}\cdot x\rangle$$

mit $|x - \ln\left|\frac{b+\sqrt{\Delta}}{b-\sqrt{\Delta}}\right|| \leq 2^{-p/2-\log p}$.

Sei $n = \operatorname{size}(\Delta)$. Der Algorithmus benötigt für die Ein-/Ausgabe höchstens $n+p+k+2$ Qubits. Zusätzlich werden für die internen Berechnungen höchstens $1,5n + p^2 + \log p + 9$ Qubits, falls die Quantenaddition verwendet wird, oder höchstens $2,5 + p^2 + p + \log p + 9$ Qubits, falls die klassische Addition verwendet wird, benötigt.

Algorithm 29 cRho-RQ

Input: $|\mathsf{ctrl}\rangle$, $|a\rangle$, $|b\rangle$, Δ, wobei $\Delta > 0$ eine Diskriminante und (a, b) ein $\mathcal{O}_\Delta$-Ideal mit $|a| \leq \Delta$ und $|b| \leq 2|a|$ ist.

Output: $|\mathsf{ctrl}\rangle$, $|a'\rangle$, $|b'\rangle$ und $|s\rangle$, wobei $(a', b') = \rho(a, b)$ und $s = \mathrm{sign}(a')\lfloor (b+q)/(2|a'|)\rfloor$, falls $\mathsf{ctrl} = 1$ gilt, und $(a', b') = (a, b)$ und $s = 0$, falls $\mathsf{ctrl} = 0$ gilt. Dabei gilt $q = \lfloor\sqrt{\Delta}\rfloor$, falls $|a'| \leq \lfloor\sqrt{\Delta}\rfloor$, und $q = |a'|$, falls $|a'| > \lfloor\sqrt{\Delta}\rfloor$.

1. Berechne $a' = (b^2 - \Delta)/(4a)$

$$|a\rangle, |b\rangle, |-\Delta\rangle \xrightarrow{\text{Mult}} |a\rangle, |b\rangle, |b^2 - \Delta\rangle \xrightarrow{\text{Div}} |a\rangle, |b\rangle, |a'\rangle$$

2. Wenn $\mathsf{ctrl} = 0$, setze $b_{new} = -b$ und vertausche a' und a.

$$|\mathsf{ctrl}\rangle, |b\rangle \xrightarrow{\text{cSub}} |\mathsf{ctrl}\rangle, |(-1)^{1-\mathsf{ctrl}} b\rangle$$
$$|\mathsf{ctrl}\rangle, |a\rangle, |a'\rangle \xrightarrow{\text{cSwap}} |\mathsf{ctrl}\rangle, |a_{new}\rangle, |a'_{new}\rangle$$

3. Lösche a

$$|a\rangle, |a'\rangle, |b\rangle \xrightarrow{\text{Mult}} |4aa'\rangle, |a'\rangle, |b\rangle \xrightarrow{\text{Mult}} |\underbrace{4aa' - b^2}_{=-\Delta}\rangle, |a'\rangle, |b\rangle \longrightarrow |0\rangle, |a'\rangle, |b\rangle$$

4. Berechne $s = \mathrm{sign}(a')\lfloor (b+q)/(2|a'|)\rfloor$, wobei $q = \lfloor\sqrt{\Delta}\rfloor$, falls $|a'| \leq \lfloor\sqrt{\Delta}\rfloor$, und $q = |a'|$, falls $|a'| > \lfloor\sqrt{\Delta}\rfloor$

$$|a'\rangle, |0\rangle, |0\rangle \xrightarrow{\text{Abs}} ||a'|\rangle, |\operatorname{sgn}(a')\rangle, |0\rangle \xrightarrow[\text{cNOT}]{|a'|>\lfloor\sqrt{\Delta}\rfloor} ||a'|\rangle, |\operatorname{sgn}(a')\rangle, |t\rangle$$
$$|t\rangle, ||a'|\rangle, |b\rangle, |0\rangle \xrightarrow[\text{cAdd}]{t=1} ||a'|\rangle, |b + t\cdot|a'|\rangle, |0\rangle \xrightarrow[\text{cAdd}]{t=0} ||a'|\rangle, |b+q\rangle, |0\rangle \xrightarrow{\text{Div}}$$
$$\xrightarrow{\text{Div}} |t\rangle, ||a'|\rangle, |b+q \mod 2|a'|\rangle, |\lfloor (b+q)/(2|a'|)\rfloor\rangle$$
$$|\operatorname{sgn}(a')\rangle, \left|\left\lfloor\frac{b+q}{2|a'|}\right\rfloor\right\rangle, |0\rangle, |0\rangle \xrightarrow{2\times\text{Copy}} |\operatorname{sgn}(a')\rangle, \left|\left\lfloor\frac{b+q}{2|a'|}\right\rfloor\right\rangle, |\operatorname{sgn}(a')\rangle, \left|\left\lfloor\frac{b+q}{2|a'|}\right\rfloor\right\rangle \xrightarrow{\text{Abs}^\dagger}$$
$$\xrightarrow{\text{Abs}^\dagger} |\operatorname{sgn}(a')\rangle, |\lfloor b + q/(2|a'|)\rfloor\rangle, |0\rangle, |s\rangle$$
$$|t\rangle, ||a'|\rangle, |b + |a'| \mod 2|a'|\rangle, |\lfloor (b+q)/(2|a'|)\rfloor\rangle \xrightarrow[2\times\text{cSub}]{\text{Mult}} |t\rangle, ||a'|\rangle, |b\rangle, |0\rangle$$
$$||a'|\rangle, |\operatorname{sgn}(a')\rangle, |t\rangle \xrightarrow[\text{cNOT}]{|a'|>\lfloor\sqrt{\Delta}\rfloor} ||a'|\rangle, |\operatorname{sgn} a'\rangle|0\rangle \xrightarrow{\text{Abs}^\dagger} |a'\rangle, |0\rangle, |0\rangle$$

5. Berechne $b' = -b + 2sa'$

$$|a'\rangle, |b\rangle, |s\rangle \xrightarrow{\text{Sub}} |a'\rangle, |-b\rangle, |s\rangle \xrightarrow{\text{Mult}} |a'\rangle, |b'\rangle, |s\rangle$$

6. Gebe aus: $|\mathsf{ctrl}\rangle$, $|a'\rangle$, $|b'\rangle$ und $|s\rangle$

Algorithm 30 cDistance

Input: $|\mathsf{ctrl}\rangle$, $|b\rangle$, $|d\rangle$, Δ, $\delta = \lceil\sqrt{\Delta}\rfloor_p$, p, k, so daß gilt: $p > \max\{64, \log\Delta\}$ und $d \in 2^{-p}\mathbb{Z}$ mit $\lfloor|d|\rfloor < 2^k < 2^5\Delta$.

Output: $|\mathsf{ctrl}\rangle$, $|b\rangle$, $|d+x\rangle$ mit $x = 0$, falls $\mathsf{ctrl} = 0$ gilt, und $x \in \mathbb{Q}$ mit $\left|x - \ln\left|\frac{b+\sqrt{\Delta}}{b-\sqrt{\Delta}}\right|\right| \leq 2^{-p/2-\log p}$, falls $\mathsf{ctrl} = 1$ gilt.

1. Unterscheide zwei Fälle $b < 0$ und $b \geq 0$

$$|0\rangle, |b\rangle \xrightarrow{\textsc{Abs}} |\operatorname{sgn}(b)\rangle, ||b|\rangle$$

2. Wenn $b < 0$, berechne das neue d

$$||b|\rangle, |-\Delta\rangle, |0\rangle \xrightarrow{\textsc{Mult}} ||b|\rangle, |b^2 - \Delta = 4ac\rangle, |0\rangle \xrightarrow{\textsc{Abs}} ||b|\rangle, ||4ac|\rangle, |\operatorname{sgn}(4ac)\rangle$$

$$|\mathsf{ctrl}\rangle, |\operatorname{sgn}(b)\rangle, ||4ac|\rangle, |d\rangle \xrightarrow[\textsc{Toffoly, cLn}_{\textsc{Add}}]{\operatorname{sgn}(b)=0,\ \mathsf{ctrl}=1} |\mathsf{ctrl}\rangle, |\operatorname{sgn}(b)\rangle, ||4ac|\rangle, |d + \mathsf{ctrl}\cdot\ln'(|4ac|)\rangle$$

$$||b|\rangle, |\operatorname{sgn}(4ac)\rangle, ||4ac|\rangle \xrightarrow[\textsc{Mult}]{\textsc{Abs}^\dagger} |b\rangle, |0\rangle, |-\Delta\rangle$$

$$||b|\rangle \xrightarrow{\textsc{Add}} ||b| + \delta\rangle$$

$$|\mathsf{ctrl}\rangle, ||b| + \delta\rangle, |d\rangle \xrightarrow[2\times\textsc{cLn}_{\textsc{Sub}}]{\operatorname{sgn} b=0,\ \mathsf{ctrl}=1} |\mathsf{ctrl}\rangle, ||b| + \delta\rangle, |d + \mathsf{ctrl}\cdot 2\ln'(b+\delta)\rangle$$

$$|\operatorname{sgn}(b)\rangle, ||b| + \delta\rangle \xrightarrow[\textsc{Abs}^\dagger]{\textsc{Sub}} |0\rangle, |b\rangle$$

3. Wenn $b \geq 0$, berechne das neue d

$$||b|\rangle, |-\Delta\rangle, |0\rangle \xrightarrow{\textsc{Mult}} ||b|\rangle, |b^2 - \Delta = 4ac\rangle, |0\rangle \xrightarrow{\textsc{Abs}} ||b|\rangle, ||4ac|\rangle, |\operatorname{sgn}(4ac)\rangle$$

$$|\mathsf{ctrl}\rangle, |\operatorname{sgn}(b)\rangle, ||4ac|\rangle, |d\rangle \xrightarrow[\textsc{Toffoly, cLn}_{\textsc{Sub}}]{\operatorname{sgn}(b)=1,\ \mathsf{ctrl}=1} |\mathsf{ctrl}\rangle, |\operatorname{sgn}(b)\rangle, ||4ac|\rangle, |d - \mathsf{ctrl}\cdot\ln'(|4ac|)\rangle$$

$$||b|\rangle, |\operatorname{sgn}(4ac)\rangle, ||4ac|\rangle \xrightarrow[\textsc{Mult}]{\textsc{Abs}^\dagger} |b\rangle, |0\rangle, |-\Delta\rangle$$

$$||b|\rangle \xrightarrow{\textsc{Add}} ||b| + \delta\rangle$$

$$|\mathsf{ctrl}\rangle, ||b| + \delta\rangle, |d\rangle \xrightarrow[2\times\textsc{cLn}_{\textsc{Add}}]{\operatorname{sgn} b=1,\ \mathsf{ctrl}=1} |\mathsf{ctrl}\rangle, ||b| + \delta\rangle, |d + \mathsf{ctrl}\cdot 2\ln'(b+\delta)\rangle$$

$$|\operatorname{sgn}(b)\rangle, ||b| + \delta\rangle \xrightarrow[\textsc{Abs}^\dagger]{\textsc{Sub}} |0\rangle, |b\rangle$$

4. Gebe aus: $|\mathsf{ctrl}\rangle$, $|b\rangle$, $|d+x\rangle$

Algorithm 31 REDUCE-RQ

Input: $|\mathsf{ctrl}\rangle$, $|a\rangle$, $|b\rangle$, $|d\rangle$, $\Delta > 0$, p, mit $p > \max\{64, \log\Delta\}$, $\delta = \lceil\sqrt{\Delta}\rfloor_p$, $\delta' = \lfloor\sqrt{\Delta}\rfloor$, wobei (a, b) ein normales $\mathcal{O}_\Delta$-Ideal mit $0 \leq a \leq \Delta$ und $d \in 2^{-p}\mathbb{Z}$, mit $\lfloor|d|\rfloor < 2^k < 2^5\Delta$ ist.
Output: $|\mathsf{ctrl}\rangle$, $|a'\rangle$, $|b'\rangle$, $|d + \mathsf{ctrl}\cdot x\rangle$, $|\mathsf{temp}\rangle$, wobei (a', b') das erste reduzierte Ideal in der Reduktionssequenz von (a, b) und $|x - 2\operatorname{Log}\gamma| \leq 2^{-p/2}$ gilt, wobei γ ein Relativerzeuger von (a', b') relativ zu (a, b) ist.

1. Initialisiere $|p\rangle$, $|\mathsf{counter}\rangle$ mit Null und $|q\rangle$ mit Eins. Dabei ist $\text{size}(p) = (\log\Delta)/2 + 2$, $\text{size}(q) = (3/2)\log\Delta + 2$ und $\text{size}(\mathsf{counter}) = \log_2\log_2\Delta - 1$.

2. Falls $a > \delta'$, setze $\mathsf{state} = 1$. Sonst setze $\mathsf{state} = 0$.

$$|a\rangle, |0\rangle \xrightarrow[\text{CNOT}]{a>\delta'} |a\rangle, |\mathsf{state}\rangle$$

3. Wiederhole $\lfloor(\log\Delta)/4 + 2\rfloor$ mal

   (a) Falls $\mathsf{state} = 1$, berechne $\rho(a, b)$ und, falls zusätzlich $\mathsf{ctrl} = 1$, berechne die neue Distanz $d_{new} = d + \mathsf{ctrl}\cdot\ln\left|\frac{b+\delta}{b-\delta}\right|$

$$|\mathsf{state}\rangle, |\mathsf{ctrl}\rangle, |0\rangle \xrightarrow{\text{TOFFOLY}} |\mathsf{state}\rangle, |\mathsf{ctrl}\rangle, |t\rangle$$
$$|\mathsf{state}\rangle, |t\rangle, |a\rangle, |b\rangle, |0\rangle, |d\rangle \xrightarrow[\text{cRHO-RQ}]{\text{cDISTANCE}} |\mathsf{state}\rangle, |\mathsf{ctrl}\rangle, |a_{new}\rangle, |b_{new}\rangle, |s\rangle, |d_{new}\rangle$$
$$|\mathsf{state}\rangle, |\mathsf{ctrl}\rangle, |t\rangle \xrightarrow{\text{TOFFOLY}} |\mathsf{state}\rangle, |\mathsf{ctrl}\rangle, |0\rangle$$

   (b) Falls $\mathsf{state} = 1$, berechne $p' = q$, $q' = p + sq$ und lösche s

$$|\mathsf{state}\rangle, |p\rangle, |q\rangle, |s\rangle, |0\rangle \xrightarrow{\text{cUNCOMPUTES}} |\mathsf{state}\rangle, |p_{new}\rangle, |q_{new}\rangle, |0\rangle, |\operatorname{sgn} p\rangle$$

   (c) Falls zum ersten Mal $a \leq \delta'$, setze $\mathsf{state} = 0$

$$|\mathsf{counter}\rangle, |a_{new}\rangle, |\mathsf{state}\rangle \xrightarrow[\text{TOFFOLY}]{\mathsf{counter}=0, a_{new}\leq\delta'} |\mathsf{counter}\rangle, |a_{new}\rangle, |\mathsf{state}_{new}\rangle$$

   (d) Falls $\mathsf{state} = 0$, erhöhe $\mathsf{counter}$ um Eins (dieser Schritt ist für Reversibilität notwendig).

$$|\mathsf{state}\rangle, |\mathsf{counter}\rangle \xrightarrow[\text{cADD}]{\mathsf{state}=0} |\mathsf{state}\rangle, |\mathsf{counter}_{new}\rangle$$

4. Damit (a, b) mit Sicherheit reduziert ist, berechne $\rho(a, b)$ aus und, falls $\mathsf{ctrl} = 1$, berechne dabei die neue Distanz

$$|1\rangle, |\mathsf{ctrl}\rangle, |a\rangle, |b\rangle, |d\rangle, |0\rangle \xrightarrow[\text{cRHO-RQ}]{\text{cDISTANCE}} |1\rangle, |\mathsf{ctrl}\rangle, |a_{new}\rangle, |b_{new}\rangle, |d_{new}\rangle, |s\rangle$$

5. Gebe aus: $|\mathsf{ctrl}\rangle$, $|a\rangle$, $|b\rangle$, $|d\rangle$, $|p, q, \mathsf{counter}, s, (\text{sgn}(p))_1, \ldots, \text{sgn}(p)_{\lfloor(\log\Delta)/4+2\rfloor}\rangle$

Die Anzahl der elementaren Quantengatter ist $O(n^3+p^3)$, falls die klassische Addition verwendet wird, bzw. $O(n^4+p^4)$, falls die Quantenaddition verwendet wird.

Beweis. Wir verwenden die gleichen Bezeichnungen wie im Satz.

Bei der Berechung des Abstands unterscheiden wir die folgenden zwei Fälle:

- Falls $b<0$, dann gilt $\ln\left|\frac{b+\sqrt{\Delta}}{b-\sqrt{\Delta}}\right| = \ln\left|\frac{b^2-\Delta}{(b-\sqrt{\Delta})^2}\right| = \ln|4ac| - 2\ln(|b|+\sqrt{\Delta})$
- Falls $b\geq 0$, dann gilt $\ln\left|\frac{b+\sqrt{\Delta}}{b-\sqrt{\Delta}}\right| = \ln\left|\frac{(b+\sqrt{\Delta})^2}{b^2-\Delta}\right| = 2\ln(b+\sqrt{\Delta}) - \ln|4ac|$

Wir zeigen nun, daß der Algorithmus, im Fall $\mathsf{ctrl}=1$, $d+x$ mit $|x-\ln\left|\frac{b+\sqrt{\Delta}}{b-\sqrt{\Delta}}\right|| \leq 2^{-p/2}$ berechnet. Bei Berechnung des Logarithmus verwenden wir cLn$_{\text{Add}}$ bzw. cLn$_{\text{Sub}}$ und setzen $n\leftarrow k$, $m\leftarrow p$ und $q\leftarrow\lceil p/2+\log p+2\rceil$. Daraus folgt

$$q+\lceil\log(q+3)\rceil+\log(n+m) \leq \left\lceil\frac{p}{2}+\log p+2\right\rceil+\lceil\log p\rceil+\log(2^6p) \leq \frac{p}{2}+3\log p+10 \leq p,$$

für $p>64$. Deshalb ist die Vorbedingung vom Satz 3.15.3 erfüllt.

Sei $b<0$, dann folgt aus Satz 3.15.3

$$\begin{aligned}
|x-\ln\left|\frac{b+\sqrt{\Delta}}{b-\sqrt{\Delta}}\right|| &= \left|x-\left(\ln|4ac|-2\ln(|b|+\lceil\sqrt{\Delta}\rfloor_p)\right)+2\left(\ln(|b|+\sqrt{\Delta})-\ln(|b|+\lceil\sqrt{\Delta}\rfloor_p)\right)\right| \\
&\leq 3(2^{-q})+2\left|\ln(|b|+\lceil\sqrt{\Delta}\rfloor_p+2^{-p})-\ln(|b|+\lceil\sqrt{\Delta}\rfloor_p)\right| \\
&\leq 2^{-p/2-\log p-1}+2|\ln(1+2^{-p})| = 2^{-p/2-\log p-1}+2\sum_{i=1}^{\infty}(-1)^{i-1}(2^{-p})^i/i \\
&\leq 2^{-p/2-\log p-1}+2^{-p+2} \leq 2^{p/2-\log p}.
\end{aligned}$$

Für den Fall $b>0$ läßt sich die Ungleichung analog beweisen.

Die Anzahl der Qubits, die für die Ein-/Ausgabe benötigt werden, ist $\text{size}(b)+\text{size}(d)+\text{size}(\mathsf{ctrl}) \leq (n+1)+(p+q+1)+1 = n+p+q+2$. Die Anzahl der Qubits, die für die internen Berechnungen verwendet werden, sowie die Anzahl der Gatter können mit den Ergebnissen aus dem Kapitel 3 einfach berechnet werden. □

Satz 5.1.3 *Seien Δ eine Diskriminante eines reell-quadratischen Zahlkörpers, (a,b) ein normales $\mathcal{O}_\Delta$-Ideal mit $0\leq a\leq\Delta$, $k,p\in\mathbb{N}$, mit $p>\max\{64,\log\Delta\}$, $d\in 2^{-p}\mathbb{Z}$ mit $\lfloor|d|\rfloor\leq 2^k\leq 2^5\Delta$ und $\delta=\lceil\sqrt{\Delta}\rfloor_p$. Reduce-RQ führt die folgende Quantenoperation durch:*

$$|\mathsf{ctrl}\rangle,|a\rangle,|b\rangle,|d\rangle,|0\rangle,\Delta,p,\delta,\lfloor\sqrt{\Delta}\rfloor \;\longrightarrow\; |\mathsf{ctrl}\rangle,|a'\rangle,|b'\rangle,|d+\mathsf{ctrl}\cdot x\rangle,|\mathsf{temp}\rangle,$$

wobei (a',b') das erste reduzierte Ideal in der Reduktionssequenz von (a,b) und $|x-2\,\text{Log}\,\gamma|\leq 2^{-p/2}$ gilt, wobei γ ein Relativerzeuger von (a',b') relativ zu (a,b) ist.

Sei $n=\text{size}(\Delta)$. Der Algorithmus benötigt $3,25n+\log_2 n+p+k+10$ Qubits für die Ein- und Ausgabe. Außerdem werden höchstens $1,75n+p^2+\log_2 p+11$ Qubits für die internen

Berechnungen verwendet, falls die Quantenaddition verwendet wird, bzw. höchstens $2,75n + p^2 + p + \log_2 p + 11$ *Qubits, falls die klassische Addition verwendet wird.*

Die Anzahl der elementaren Quantengatter ist $O(n^4 + np^3)$*, falls die klassische Addition verwendet wird, bzw.* $O(n^5 + np^4)$*, falls die Quantenaddition verwendet wird.*

Beweis. Wir verwenden die Bezeichnungen wie im Algorithmus und im Satz.

Als erstes zeigen wir, daß der Algorithmus ein reduziertes Ideal in der Äquivalenzklasse des Eingabe-Ideals berechnet. Wegen $a \leq \Delta$ folgt aus dem Satz 2.5.52, daß die Anzahl der Reduktionsschritte höchstens $\frac{1}{2}\log_2(|a|/\sqrt{\Delta})+2 \leq (\log\Delta)/4+2$ ist. Deshalb ist die Anzahl der Wiederholungen im Schritt 3 ausreichend. Solange $a > \delta'$ gilt, wird der Algorithmus cRho-RQ angewendet und die Distanz mit dem Algorithmus cDistance berechnet. Außerdem folgt aus dem Satz 2.5.59, daß $|s| > 1$ ist. Deshalb ist die Vorbedingung von cUncomputeS erfüllt. Wenn $a \leq \delta'$ zum ersten Mal auftritt, wird state auf Null gesetzt, so daß ab diesem Moment in der Schleife keine Reduktionsschritte mehr durchgeführt werden. Gilt $a \leq \delta'$, dann folgt aus dem Satz 2.5.53, daß nach höchstens einem Schritt das Ideal reduziert ist. Dies wird im Schritt 4 durchgeführt. Die Anzahl der Reduktionsschritte wird durch das Register state kontrolliert.

Als nächstes zeigen wir, daß die berechnete Distanz $|x - 2\operatorname{Log}\gamma| \leq 2^{-p/2}$ gilt. Bei der Distanzberechnung führen wir den Algorithmus cDistance höchstens $\lfloor(\log\Delta)/4+2\rfloor$ mal aus. Bei jeder Ausfühurung summieren sich die Approximationsfehler zu einem Gesamtfehler

$$2^{-p/2-\log p}\lfloor(\log\Delta)/4+2\rfloor \leq 2^{-p/2}p^{-1}(\frac{p}{4}+2) \leq 2^{-p/2}.$$

Für die Eingabe benötigt der Algorithmus

$$\operatorname{size}(a)+\operatorname{size}(b)+\operatorname{size}(d)+1 \leq 2n+p+k+2$$

Qubits. Für die Ausgabe benötigt der Algorithmus

$$1+\operatorname{size}(a')+\operatorname{size}(b')+\operatorname{size}(p)+\operatorname{size}(q)+\operatorname{size}(\mathsf{counter})+\operatorname{size}(s)+\lfloor(\log\Delta)/4+2\rfloor \leq 6,25n+\log n+10$$

Qubits.

Die Anzahl der Qubits, die für die internen Berechnungen notwendig sind, sowie die Anzahl der notwendigen Gatter lassen sich einfach mit den Ergebnissen aus dem Kapitel 3 berechnen. □

5.2 Berechnung des rechten und linken Nachbarn

In diesem Abschnitt präsentieren wir Algorithmen zum Berechnen von Nachbarn eines reduzierten Ideals. Die Algorithmen sind auf Seite 87 und 88 abgebildet. Ihre Laufzeit und Speicherplatzbedarf werden in den folgenden Sätzen untersucht.

Satz 5.2.1 *Sei Δ eine Diskriminante eines reell-quadratischen Zahlkörpers und (a,b) ein reduziertes $\mathcal{O}_\Delta$-Ideal. Der Algorithmus* InverseRho *führt die folgende Quantenoperation durch:*

$$|a_1\rangle, |b_1\rangle, |0\rangle, \Delta, \lfloor\sqrt{|\Delta|/3}\rfloor \longrightarrow |a'\rangle, |b'\rangle, |t\rangle, \Delta, \lfloor\sqrt{|\Delta|/3}\rfloor,$$

wobei (a',b') ein reduziertes $\mathcal{O}_\Delta$-Ideal ist und $\rho(a',b') = (a,b)$ und $b' = -b + 2ta$ gilt.

Sei $n = \mathrm{size}(\Delta)$. Der Algorithmus benötigt für die Ein-/Ausgabe höchstens $1.5n + 4$ Qubits. Zusätzlich werden für die internen Berechnungen höchstens $n+7$ Qubits, falls die Quantenaddition verwendet wird, und höchstens $1,5n+6$ Qubits, falls die klassische Addition verwendet wird, benötigt.

Die Anzahl der elementaren Quantengatter ist $O(n^2)$, falls die klassische Addition verwendet wird, bzw. $O(n^3)$, falls die Quantenaddition verwendet wird.

□

Satz 5.2.2 *Seien Δ eine Diskriminante eines reell-quadratischen Zahlkörpers, (a,b) ein reduziertes $\mathcal{O}_\Delta$-Ideal, $p \in \mathbb{N}$ und $d \in 2^{-p}\mathbb{Z}$, $|d| \leq \Delta$. Der Algorithmus* cRightNeighbour *führt die folgende Quantenoperation durch:*

$$|\mathsf{ctrl}\rangle, |a\rangle, |b\rangle, |d\rangle, \Delta, p, \lceil\sqrt{\Delta}\rfloor_p, \lfloor\sqrt{|\Delta|/3}\rfloor \longrightarrow |\mathsf{ctrl}\rangle, |a'\rangle, |b'\rangle, |d'\rangle,$$

wobei (a',b') der rechte Nachbar von (a,b) ist und $x = 0$ ist, falls $\mathsf{ctrl} = 0$ bzw. $\left|x - \ln\left|\frac{b+\delta}{b-\delta}\right|\right| \leq 2^{-p/2-\log p}$ gilt, falls $\mathsf{ctrl} = 1$ ist.

Sei $n = \mathrm{size}(\Delta)$. Der Algorithmus benötigt für die Ein-/Ausgabe höchstens $n+p+k+4$ Qubits. Zusätzlich werden für die internen Berechnungen höchstens $1,5n + \max\{n, p^2 + logp + 10\}$ Qubits, falls die Quantenaddition verwendet wird, bzw. höchstens $2,5n + \max\{0, 5n, p^2 + p + logp + 10\}$ Qubits, falls die klassische Addition verwendet wird, benötigt.

Die Anzahl der elementaren Quantengatter ist $O(n^3 + p^3)$, falls die klassische Addition verwendet wird, bzw. $O(n^4 + p^4)$, falls die Quantenaddition verwendet wird. □

Algorithm 32 InverseRho

Input: $|a\rangle$, $|b\rangle$, Δ, $\lfloor\sqrt{\Delta}\rfloor$, wobei $\Delta > 0$ eine Diskriminante und (a, b) ein reduziertes $\mathcal{O}_\Delta$-Ideal ist.
Output: $|a'\rangle$, $|b'\rangle$, $|t\rangle$, wobei $(a, b) = \rho(a', b')$ gilt und s die Gleichung $b' = -b + 2ta'$ erfüllt.

1. Berechne $t = \operatorname{sign}(a) \left\lfloor (b + \lfloor\sqrt{\Delta}\rfloor)/(2|a|) \right\rfloor$

$$|a\rangle, |b\rangle, |0\rangle, |0\rangle \xrightarrow{\text{Abs}} ||a|\rangle, |b\rangle, |\operatorname{sign}(a)\rangle, |0\rangle \xrightarrow{\text{Add}} ||a|\rangle, |b + \lfloor\sqrt{\Delta}\rfloor\rangle, |\operatorname{sign}(a)\rangle, |0\rangle \xrightarrow{\text{Div}}$$
$$\xrightarrow{\text{Div}} ||a|\rangle, |b + \lfloor\sqrt{\Delta}\rfloor \mod 2|a|\rangle, |\operatorname{sign}(a)\rangle, |t'\rangle$$
$$|\operatorname{sign}(a)\rangle, |t'\rangle, |0\rangle \xrightarrow{\text{cSub}} |\operatorname{sign}(a)\rangle, |t'\rangle, |t\rangle$$
$$||a|\rangle, |b + \lfloor\sqrt{\Delta}\rfloor \mod 2|a|\rangle, |\operatorname{sign}(a)\rangle, |t'\rangle \xrightarrow[\text{Sub,Abs}^\dagger]{\text{Mult}} |a\rangle, |b\rangle, |0\rangle, |0\rangle$$

2. Berechne $b' = -b + 2ta$

$$|a\rangle, |b\rangle, |t\rangle \xrightarrow{\text{Sub}} |a\rangle, |-b\rangle, |t\rangle \xrightarrow{\text{Mult}} |a\rangle, |b'\rangle, |t\rangle$$

3. Berechne $a' = (b^2 - \Delta)/(4a)$

$$|a\rangle, |b\rangle, |-\Delta\rangle \xrightarrow{\text{Mult}} |a\rangle, |b\rangle, |b^2 - \Delta\rangle \xrightarrow{\text{Div}} |a\rangle, |b\rangle, |a'\rangle$$

4. Lösche a

$$|a\rangle, |a'\rangle, |b\rangle \xrightarrow{\text{Mult}} |4aa'\rangle, |a'\rangle, |b\rangle \xrightarrow{\text{Mult}} |\underbrace{4aa' - b^2}_{=-\Delta}\rangle, |a'\rangle, |b\rangle \longrightarrow |0\rangle, |a'\rangle, |b\rangle$$

5. Gebe aus: $|a'\rangle$, $|b'\rangle$ und $|t\rangle$.

Algorithm 33 cRightNeighbour

Input: $|\mathsf{ctrl}\rangle$, $|a\rangle$, $|b\rangle$, $|d\rangle$, Δ, $p \in \mathbb{N}$, $\delta = \lceil\sqrt{\Delta}\rfloor_p$ und $\lfloor\sqrt{\Delta}\rfloor$, wobei $\Delta > 0$ eine Diskriminante, (a, b) ein reduziertes $\mathcal{O}_\Delta$-Ideal und $d \in 2^{-p}\mathbb{Z}$, $|d| < \Delta$ ist.
Output: $|\mathsf{ctrl}\rangle$, $|a'\rangle$, $|b'\rangle$, $|d + \mathsf{ctrl} \cdot x\rangle$, wobei (a', b') der rechte Nachbar von (a, b) ist und $x = 0$ ist, falls $\mathsf{ctrl} = 0$ bzw. $\left|x - \ln\left|\frac{b+\delta}{b-\delta}\right|\right| \leq 2^{-p/2-\log p}$ gilt, falls $\mathsf{ctrl} = 1$ ist.

1. Berechne die Distanz zwischen (a, b) und (a', b')

$$|\mathsf{ctrl}\rangle, |b\rangle, |\mathsf{dist}\rangle \xrightarrow{\text{cDistance}} |\mathsf{ctrl}\rangle, |b\rangle, \left|d + \mathsf{ctrl} \cdot \ln\left|\frac{b+\delta}{b-\delta}\right|\right\rangle$$

2. Falls $\mathsf{ctrl} = 1$, berechne (a', b'), den rechten Nachbarn von (a, b)

$$|a\rangle, |b\rangle, |0\rangle \xrightarrow{\text{cRho-RQ}} |a'\rangle, |b'\rangle, |s\rangle$$

3. Kopiere a' und b'

$$|a'\rangle, |b'\rangle, |0\rangle, |0\rangle \xrightarrow{2\times\text{Copy}} |a'\rangle, |b'\rangle, |a'\rangle, |b'\rangle$$

4. Führe Schritt 2 rückwärts aus

$$|a'\rangle, |b'\rangle, |s\rangle \xrightarrow{\text{cRho-RQ}^\dagger} |a\rangle, |b\rangle, |0\rangle$$

5. Um a und b zu löschen, berechne (a'', b''), den linken Nachbarn von (a', b')

$$|a'\rangle, |b'\rangle, |0\rangle \xrightarrow{\text{InverseRho}} |a''\rangle, |b''\rangle, |t\rangle$$

6. Es gilt $a'' = a$ und $b'' = b$. Lösche a und b durch das Kopieren.

$$|a''\rangle, |a\rangle, |b''\rangle, |b\rangle \xrightarrow{2\times\text{Copy}} |a''\rangle, |0\rangle, |b''\rangle, |0\rangle$$

7. Führe Schritt 5 rückwärts aus um a'', b'' und t zu löschen

$$|a''\rangle, |b''\rangle, |t\rangle \xrightarrow{\text{InverseRho}^\dagger} |a'\rangle, |b'\rangle, |0\rangle$$

8. Gebe aus: $|\mathsf{ctrl}\rangle$, $|a'\rangle$, $|b'\rangle$, $\left|\mathsf{dist} + \mathsf{ctrl} \cdot \ln\left|\frac{b+\delta}{b-\delta}\right|\right\rangle$

5.3 Multiplikation und Reduktion von Idealen

In diesem Abschnitt entwickeln wir einen Algorithmus, welcher die folgende Quantenoperation durchführt:

$$|\mathsf{ctrl}\rangle, |a_1\rangle, |b_1\rangle, |d_1\rangle, |0\rangle, a_2, b_2, d_2, \Delta, p \longrightarrow |\mathsf{ctrl}\rangle, |a\rangle, |b\rangle, |d\rangle, |\mathsf{temp}\rangle, \tag{5.1}$$

wobei (a_1, b_1), (a_2, b_2) und (a, b) reduzierte $\mathcal{O}_\Delta$-Ideale und $d_1, d_2, d \in 2^{-p} \in \mathbb{Z}$ sind, so daß gilt:

- $0 < d_1, d_2 \leq \ln \Delta$, so daß $d_i + y_i < 0$, $i = 1, 2$, wobei y_i die Approximation des Abstands zwischen (a, b) und $\rho(\rho(a, b))$ ist. Diese Approximation ist dabei eindeutig durch die Wahl von Ln, cDistance und cRightNeighbour bestimmt,
- $(a, b) = (a_1, b_1)$ und $d_1 = d$, falls $\mathsf{ctrl} = 0$, bzw.
- (a, b) ist dasjenige reduzierte Ideal in der Idealklasse von $(a_1, b_1) \cdot (a_2, b_2)$, für welches die Ungleichungen $a > 0$, $d_1 + d_2 + x < 0$ und $d_1 + d_2 + x + y > 0$ erfüllt sind, wobei x die Approximation des Abstands zwischen $(a_1, b_1) \cdot (a_2, b_2)$ und (a, b) ist und y ist die Approximation des Abstands zwischen (a, b) und $\rho(\rho(a, b))$ ist. Die Approximationen sind dabei eindeutig durch die Wahl von Ln, cDistance, Reduce-RQ und cRightNeighbour bestimmt.

Wir beschreiben kurz die Idee des Algorithmus. Im ersten Schritt berechnen wir das neue Ideal (a, b) und die neue Distanz d. Im zweiten Schritt löschen wir das Ideal (a_1, b_1). Dieses Löschen ist jedoch nicht komplett, denn wegen der ungenauen Logarithmusberechnungen kann das Ideal (a_1, b_1) nicht aus den Idealen (a, b) und (a_2, b_2) exakt bestimmt werden. Im unten präsentierten Algorithmus stellen wir eine Lösung vor, die mit zwei zusätzlichen Qubits das Ideal (a_1, b_1) eindeutig bestimmt, so daß es gelöscht werden kann. Diese Qubits müssen jedoch gespeichert bleiben, damit der Algorithmus reversibel ist.

Um das Ideal (a, b) zu bestimmen, gehen wir folgendermaßen vor: Im Fall $\mathsf{ctrl} = 1$ bilden wir als erstes das Produkt der Ideale (a_1, b_1) und (a_2, b_2) und reduzieren sie. Gleichzeitig addieren wir d_1, d_2 und die Abstände, die während der Reduktion berechnet werden. Danach gehen wir nach rechts auf dem Reduktionszyklus[1] und addieren dabei die Abstände zum Register $|d\rangle$. Nach $\lfloor 6 \ln \Delta \rfloor$ Schritten, gilt $d < 0$ (das folgt aus Sätzen 2.5.72 und 2.5.73). Nun gehen wir $\lfloor 6 \ln \Delta + 5 \rfloor$ Schritte nach links auf dem Reduktionszyklus. Wenn dabei zum ersten Mal ein Ideal (a, b) mit $a > 0$ gefunden wird und gleichzeitig $d > 0$ gilt, kopieren wir dieses Ideal in die Ausgaberegister. Ab diesem Zeitpunkt werden keine Abstandsberechnungen mehr durchgeführt. Zum Löschen der temporären Register gehen wir schließlich fünf Schritte nach rechts auf dem Reduktionszyklus und führen die Reduktion und Multiplikation rückwärts aus.

Im Fall $\mathsf{ctrl} = 0$ werden die gleichen Berechnungen mit Idealen durchgeführt wie oben. Es werden dabei jedoch keine Abstände berechnet. In die Ausgangsregister wird das Ideal (a_1, b_1) kopiert.

[1] D.h. wir wenden den Reduktionsopereator ρ, ρ^2, ρ^3, ... auf das reduzierte Ideal

Um das Ideal (a_1, b_1) zu löschen, verfahren wir analog. Wie bereits erwähnt, läßt sich das Ideal (a_1, b_1) nicht ohne weiteres exakt bestimmen. Wir können jedoch eine Folge $(\mathfrak{a}, \rho^2(\mathfrak{a}), \rho^4(\mathfrak{a}))$ berechnen, so daß (a_1, b_1) ein Element davon ist. Mit zwei Qubits läßt sich markieren, an welcher Stelle in der Folge (a_1, b_1) steht.

Um die bessere Lesbarkeit zu erreichen, unterteilen wir den Multiplikationsalgorithmus in mehrere Teilalgorithmen, welche auf den folgenden Seiten dargestellt sind. Die Laufzeit und der Speicherplatzbedarf werden im folgenden Satz untersucht.

Satz 5.3.1 *Sei Δ eine Diskriminante eines reell-quadratischen Zahlkörpers, (a_1, b_1), (a_2, b_2) reduzierte $\mathcal{O}_\Delta$-Ideale und $p \in \mathbb{N}$. Der Algorithmus* Multiplication-RQ *berechnet die in (5.1) definierte Quantenoperation.*

Sei $n = \mathrm{size}(\Delta)$. Der Algorithmus benötigt für die Ein-/Ausgabe höchstens $n+p+k+2$ Qubits. Zusätzlich werden für die internen Berechnungen höchstens $7n + \log_2 n + 25 + \max\{n/2, p^2 + \log_2 p\}$ Qubits, falls die Quantenaddition verwendet wird, bzw. höchstens $8n + \log_2 n + 25 + \max\{n/2, p^2 + p + \log_2 p\}$ Qubits, falls die klassische Addition verwendet wird, gebraucht.

Die Anzahl der elementaren Quantengatter ist $O(n^4 + np^3)$, falls die klassische Addition verwendet wird, bzw. $O(n^5 + np^4)$, falls die Quantenaddition verwendet wird. □

Algorithm 34 MULTIPLICATION-RQ

Input: Ein Kontroll-Qubit $|\mathsf{ctrl}\rangle$, eine Diskriminante $\Delta > 0$, Zahlen $p \in \mathbb{N}$, $|a_1\rangle$, $|b_1\rangle$, $|d_1\rangle$, a_2, b_2, d_2 wie in 5.1 definiert.
Output: $|\mathsf{ctrl}\rangle$, $|a_{out}\rangle$, $|b_{out}\rangle$, $|d_{out}\rangle$, $|\mathsf{temp}\rangle$ wie in (5.1) definiert.

1. Berechne (a', b'), das Produkt der Ideale (a_1, b_1) und (a_2, b_2) und $d = d_1 + \mathsf{ctrl} \cdot d_2$

$$|\mathsf{ctrl}\rangle, |a_1\rangle, |b_1\rangle, |0\rangle, |0\rangle, |d_1\rangle \overset{\substack{\text{IDEAL-MULT} \\ \text{CADD}}}{\longrightarrow} |\mathsf{ctrl}\rangle, |a_1\rangle, |b_1\rangle, |a'\rangle, |b'\rangle, |d\rangle$$

2. Normalisiere das Ideal (a', b')

$$|a'\rangle, |b'\rangle, |0\rangle, |0\rangle, |0\rangle \overset{\text{NORMALIZE}}{\longrightarrow} |a'\rangle, |b_{new}\rangle, |s_0\rangle, |s_1\rangle, |s\rangle$$

3. Berechne das reduzierte Ideal (a, b), welches in (5.1) definiert ist. Falls $\mathsf{ctrl} = 1$, berechne zusätzlich die neue Distanz

$$|\mathsf{ctrl}\rangle, |a_1\rangle, |b_1\rangle, |a'\rangle, |b_{new}\rangle, |d\rangle, |0\rangle, |0\rangle \overset{\text{CCOMPUTENEWIDEAL}}{\longrightarrow} \longrightarrow |\mathsf{ctrl}\rangle, |a_1\rangle, |b_1\rangle, |a'\rangle, |b_{new}\rangle, |d_{new}\rangle, |a_{out}\rangle, |b_{out}\rangle$$

4. Wende die Schritte 1 und 2 rückwärts an, um die temporären Register s_0, s_1, s, a' und b_{new} zu löschen.

5. Lösche a_1 und b_1

$$|a_1\rangle, |b_1\rangle, |a_{out}\rangle, |b_{out}\rangle, |d_{new}\rangle, |0\rangle \overset{\text{CUNCOMPUTEOLDIDEAL}}{\longrightarrow} |0\rangle, |0\rangle, |a_{out}\rangle, |b_{out}\rangle, |d_{new}\rangle, |\mathsf{temp}\rangle$$

6. Gebe aus: $|\mathsf{ctrl}\rangle$, $|a_{out}\rangle$, $|b_{out}\rangle$, $|d_{new}\rangle$, $|\mathsf{temp}\rangle$.

Algorithm 35 NORMALIZE

Input: $|a\rangle$, $|b\rangle$, Δ, wobei Δ eine Diskriminante eines reell-quadratischen Zahlkörpers ist, und (a, b) ein $\mathcal{O}_\Delta$-Ideal mit $0 < a < \Delta$ und $0 < b < 2a$ ist.
Output: $|a\rangle$, $|b_{new}\rangle$, $|s_0, s_1, s\rangle$, wobei $a < b \leq 2a$, falls $a \geq \sqrt{\Delta}$, und $\sqrt{\Delta} - 2a < b < \sqrt{\Delta}$, falls $a < \sqrt{\Delta}$.

1. Berechne Hilfsvariablen $s_0, s_1 \in \{0, 1\}$, so daß $s_0 = 0$ genau dann, wenn $a \geq \sqrt{\Delta}$ und $s_1 = 0$ genau dann, wenn $a > b$

$$|a\rangle, |b\rangle, |0\rangle, |0\rangle \overset{a \geq \sqrt{\Delta}}{\longrightarrow} |a\rangle, |b\rangle, |s_0\rangle, |0\rangle \overset{a > b}{\longrightarrow} |a\rangle, |b\rangle, |s_0\rangle, |s_1\rangle$$

2. Normalisiere (a, b), falls $a \geq \sqrt{\Delta}$

$$|a\rangle, |b\rangle, |s_0\rangle \underset{\text{cSUB}}{\overset{s_0=0}{\longrightarrow}} |a\rangle, |\underbrace{b - 2s_0 a}_{=:b'}\rangle, |s_0\rangle$$

3. Normalisiere (a, b), falls $a < \sqrt{\Delta}$

$$|b'\rangle \underset{2\times\text{cSUB}}{\overset{s_0=1}{\longrightarrow}} |\lfloor\sqrt{\Delta}\rfloor - b'\rangle$$

$$|a\rangle, |\lfloor\sqrt{\Delta}\rfloor - b'\rangle, |s_0\rangle, |0\rangle \underset{\text{cDIV}}{\overset{s_0=1}{\longrightarrow}} |a\rangle, |\underbrace{(\lfloor\sqrt{\Delta}\rfloor - b') \bmod (2a)}_{=:s'}\rangle, |s_0\rangle, |\underbrace{\lfloor(\lfloor\sqrt{\Delta}\rfloor - b')/(2a)\rfloor}_{=:s}\rangle,$$

$$|a\rangle, |s'\rangle, |s_0\rangle, |s\rangle \underset{\text{cMULT}}{\overset{s_0=1}{\longrightarrow}} |a\rangle, |b\rangle, |s_0\rangle, |s\rangle$$

$$|a\rangle, |b\rangle, |s\rangle, |s_0\rangle \underset{\text{cMULT}}{\overset{s_0=1}{\longrightarrow}} |a\rangle, |b_{new}\rangle, |s_0\rangle, |s\rangle$$

4. Gebe aus: $|a\rangle$, $|b_{new}\rangle$, $|s_0, s_1, s\rangle$.

Algorithm 36 cComputeNewIdeal

Input: $|\mathsf{ctrl}\rangle$, $|a_1\rangle$, $|b_1\rangle$, $|a\rangle$, $|b\rangle$, $|d\rangle$, Δ
Output: $|\mathsf{ctrl}\rangle$, $|a_1\rangle$, $|b_1\rangle$, $|a\rangle$, $|b\rangle$, $|d_{out}\rangle$

1. Reduziere (a, b) und, falls $\mathsf{ctrl} = 1$, berechne dabei die neue Distanz

$$|\mathsf{ctrl}\rangle, |a\rangle, |b\rangle, |d\rangle, |0\rangle \overset{\text{Reduce-RQ}}{\longrightarrow} |\mathsf{ctrl}\rangle, |a_{new}\rangle, |b_{new}\rangle, |d_{new}\rangle|\mathsf{temp}\rangle$$

2. Wiederhole cRightNeighbour $\lfloor 6 \ln \Delta \rfloor$ mal, um sicherzustellen, daß $d < 0$ ist, falls $\mathsf{ctrl} = 1$

$$|\mathsf{ctrl}\rangle, |a\rangle|b\rangle, |d\rangle \overset{\text{cRightNeighbour}}{\longrightarrow} |\mathsf{ctrl}\rangle, |a_{new}\rangle, |b_{new}\rangle|d_{new}\rangle$$

3. Initialisiere $|q\rangle$: $\quad |\mathsf{ctrl}\rangle|0\rangle \overset{\text{cNOT}}{\longrightarrow} |\mathsf{ctrl}\rangle, |q\rangle$

4. Wiederhole cRightNeighbour† $\lfloor 6 \ln \Delta \rfloor + 5$ mal, damit $d > 0$ mit Sicherheit gilt. Wenn d zum ersten Mal größer oder gleich Null mit $a > 0$ ist, dann setze $q = 0$ und kopiere a, b, falls $\mathsf{ctrl} = 1$ ist, in die Ausgaberegister $|a_{out}\rangle$, $|b_{out}\rangle$. Ab diesem Zeitpunkt wird d nicht mehr verändert.

$$|q\rangle, |a\rangle, |b\rangle, |d\rangle \overset{\text{cRightNeighbour}^\dagger}{\longrightarrow} |q\rangle, |a_{new}\rangle, |b_{new}\rangle|d_{new}\rangle$$

$$|\mathsf{ctrl}\rangle, |a\rangle, |a_{out}\rangle, |d\rangle, |q\rangle, |0\rangle \overset{\substack{\mathsf{ctrl}=1, a>0, a_{out}=0 \\ d\leq 0,\ 6\times\text{Toffoly}}}{\longrightarrow} |\mathsf{ctrl}\rangle, |a\rangle, |a_{out}\rangle, |d\rangle, |q_{new}\rangle, |q'\rangle$$

$$|\mathsf{ctrl}\rangle, |q'\rangle, |a\rangle, |b\rangle, |a_{out}\rangle, |b_{out}\rangle \overset{\substack{\mathsf{ctrl}=1,\ q'=1 \\ 2\times\text{cCopy}}}{\longrightarrow} |\mathsf{ctrl}\rangle, |q'\rangle, |a\rangle, |b\rangle, |(a_{out})_{new}\rangle, |(b_{out})_{new}\rangle$$

$$|\mathsf{ctrl}\rangle, |a\rangle, |a_{out}\rangle, |b\rangle, |b_{out}\rangle, |q'\rangle \overset{\substack{\mathsf{ctrl}=1,\ a=a_{out},\ b=b_{out} \\ 3\times\text{Toffoly}}}{\longrightarrow} |\mathsf{ctrl}\rangle, |a\rangle, |a_{out}\rangle, |b\rangle, |b_{out}\rangle, |0\rangle$$

5. Falls $\mathsf{ctrl} = 0$, kopiere a_1 und b_1 in die Ausgaberegister

$$|\mathsf{ctrl}\rangle, |a_1\rangle, |a\rangle, |b_1\rangle, |b\rangle \overset{\substack{\mathsf{ctrl}=0 \\ 2\times\text{cCopy}}}{\longrightarrow} |\mathsf{ctrl}\rangle, |a_1\rangle, |a_{new}\rangle, |b_1\rangle, |b_{new}\rangle$$

6. Wende fünfmal cRightNeighbour an und führe den Schritt 1 rückwärts aus, um die Register $|a\rangle$ und $|b\rangle$ in den Ausgangszustand zu versetzen und um die temporäre Daten zu löschen.

$$|0\rangle, |a\rangle, |b\rangle, |d\rangle, |\mathsf{temp}\rangle \overset{\substack{5\times\text{cRightNeighbour} \\ \text{Reduce-RQ}^\dagger}}{\longrightarrow} |0\rangle, |a_{new}\rangle, |b_{new}\rangle, |d\rangle, |0\rangle$$

7. Gebe aus: $|\mathsf{ctrl}\rangle$, $|a_1\rangle$, $|b_1\rangle$, $|a\rangle$, $|b\rangle$, $|d\rangle$, $|a_{out}\rangle$, $|b_{out}\rangle$

Algorithm 37 cUncomputeOldIdeal

Input: $|\mathsf{ctrl}\rangle$, $|a_1\rangle$, $|b_1\rangle$, $|a\rangle$, $|b\rangle$, $|d\rangle$, Δ, a_2, b_2, d_2.
Output: $|\mathsf{ctrl}\rangle$, $|0\rangle$, $|0\rangle$, $|a\rangle$, $|b\rangle$, $|d\rangle$, $|t_1, t_2\rangle$.

1. Falls $\mathsf{ctrl} = 0$, lösche a_1 und b_1 durch Kopieren

$$|\mathsf{ctrl}\rangle, |a\rangle, |a_1\rangle, |b\rangle, |b_1\rangle \xrightarrow[2\times\text{cCopy}]{\mathsf{ctrl}=0} |\mathsf{ctrl}\rangle, |a\rangle, |0\rangle, |b\rangle, |0\rangle$$

2. Berechne (a', b'), das Produkt der Ideale (a, b) und $(a_2, -b_2)$ und, falls $\mathsf{ctrl} = 1$, die Distanz $d' = d - d_2$

$$|\mathsf{ctrl}\rangle, |a\rangle, |b\rangle, |0\rangle, |0\rangle, |d\rangle \xrightarrow[\mathsf{ctrl}=1,\text{cSub}]{\text{Ideal-Mult}} |\mathsf{ctrl}\rangle, |a\rangle, |b\rangle, |a'\rangle, |b'\rangle, |d_{new}\rangle$$

3. Normalisiere und anschließend reduziere (a', b') und, falls $\mathsf{ctrl} = 1$, berechne die neue Distanz

$$|\mathsf{ctrl}\rangle, |a'\rangle, |b'\rangle, |d\rangle, |0\rangle \xrightarrow[\text{Reduce-RQ}]{\text{Normalize}} |\mathsf{ctrl}\rangle, |a'_{new}\rangle, |b'_{new}\rangle, |d_{new}\rangle|\mathsf{temp}\rangle$$

4. Wiederhole cRightNeighbour $\lfloor 6 \ln \Delta \rfloor$ mal, um sicherzustellen, daß $d' < 0$ ist

$$|\mathsf{ctrl}\rangle, |a'\rangle|b'\rangle, |d'\rangle \xrightarrow{\text{cRightNeighbour}} |\mathsf{ctrl}\rangle, |a'_{new}\rangle, |b'_{new}\rangle|d'_{new}\rangle$$

5. Initialisiere $|q_1\rangle$ und $|t\rangle$ mit Null, $t = t_1 t_2 \in \{0, 1, 2, 3\}$

6. Wiederhole cRightNeighbour† $2 \cdot \lfloor 6 \ln \Delta \rfloor$ mal, damit $d' > 0$ mit Sicherheit gilt.

$$|\mathsf{ctrl}\rangle, |a'\rangle, |b'\rangle, |d'\rangle \xrightarrow{\text{cRightNeighbour}^\dagger} |\mathsf{ctrl}\rangle, |a'_{new}\rangle, |b'_{new}\rangle|d'_{new}\rangle$$

$$|a_1\rangle, |b_1\rangle, |a'_{new}\rangle, |b'_{new}\rangle, |d'_{new}\rangle, |q_1\rangle, |t\rangle \xrightarrow{\text{Compute-t}} |a'_1\rangle, |b'_1\rangle, |a'_{new}\rangle, |b'_{new}\rangle, |d'_{new}\rangle, |q'_1\rangle, |t'\rangle$$

7. Wende cRightNeighbour $\lfloor 6 \ln \Delta \rfloor$ mal und anschließend die Schritte 3 und 2 rückwärts an, um die temporären Register zu löschen

$$|\mathsf{ctrl}\rangle, |a\rangle, |b\rangle, |a'\rangle, |b'\rangle, |d'\rangle, |\mathsf{temp}\rangle \xrightarrow[\text{Reduce-RQ}^\dagger,\text{Normalize}^\dagger,\text{Ideal-Mult}^\dagger]{\lfloor 6 \ln \Delta \rfloor \times \text{cRightNeighbour,cSub}} |\mathsf{ctrl}\rangle, |a\rangle, |b\rangle, |d\rangle, |0\rangle$$

8. Gebe aus: $|\mathsf{ctrl}\rangle$, $|0\rangle$, $|0\rangle$, $|a\rangle$, $|b\rangle$, $|d\rangle$, $|t_1, t_2\rangle$.

Algorithm 38 COMPUTE-T

Input: $|a_1\rangle$, $|b_1\rangle$, $|a\rangle$, $|b\rangle$, $|d\rangle$, $|q_1\rangle$, $|t_1 t_1\rangle$, Δ.
Output: $|a_1'\rangle$, $|b_1'\rangle$, $|a\rangle$, $|b\rangle$, $|d\rangle$, $|q_1'\rangle$, $|t_1' t_2'\rangle$.

1. Wenn $(a_1, b_1) = (a, b)$, setze $q_1 = 1$ und $q_2 = 1$ und lösche $|a_1\rangle$ und $|b_1\rangle$

$$|a_1\rangle, |a\rangle, |b_1\rangle, |b\rangle, |q_1\rangle \xrightarrow[3\times\text{TOFFOLY}]{a_1=a, b_1=b} |a_1\rangle, |a\rangle, |b_1\rangle, |b\rangle, |q_{1,new}\rangle$$

$$|a_1\rangle, |q_1\rangle, |0\rangle \xrightarrow[\text{TOFFOLY}]{a_1\neq 0, q_1=1} |a_1\rangle, |q_1\rangle, |q_2\rangle$$

$$|q_2\rangle, |a\rangle, |a_1\rangle, |b\rangle, |b_1\rangle \xrightarrow{2\times\text{CCOPY}} |q_2\rangle, |a\rangle, |a_{1,new}\rangle, |b\rangle, |b_{1,new}\rangle$$

2. Berechne t_1 und t_2 (diese Qubits sind für die Reversibilität notwendig). Außerdem berechne q_3 und q_4

$$|d\rangle, |0\rangle \xrightarrow[\text{CNOT}]{d\leq 0} |d\rangle, |q_3\rangle$$

$$|q_2\rangle, |d\rangle, |t_1\rangle \xrightarrow[\text{TOFFOLY}]{q_2=1, d\leq 0} |q_2\rangle, |d\rangle, |t_{1,new}\rangle$$

$$|1\rangle, |a\rangle, |b\rangle, |d\rangle \xrightarrow{2\times\text{CRIGHTNEIGHBOUR}} |1\rangle, |a'\rangle, |b'\rangle, |d'\rangle$$

$$|q_2\rangle, |d'\rangle, |t_2\rangle \xrightarrow[\text{TOFFOLY}]{q_2=1, d'\leq 0} |q_2\rangle, |d'\rangle, |t_{2,new}\rangle$$

$$|d'\rangle, |0\rangle \xrightarrow[\text{CNOT}]{d'\leq 0} |d\rangle, |q_4\rangle$$

3. Lösche q_2

$$|1\rangle, |a'\rangle, |b'\rangle |d'\rangle \xrightarrow{2\times\text{CRIGHTNEIGHBOUR}} |1\rangle, |a''\rangle, |b''\rangle, |d''\rangle$$

$$|q_1\rangle, |q_2\rangle, |q_3\rangle, |q_4\rangle, |t_1\rangle, |t_2\rangle, |d''\rangle \xrightarrow[4\times\text{CNOT},\ 5\times\text{TOFFOLY}]{q_1=1, q_3=t_1, q_4=t_2, d''>0} |q_1\rangle, |0\rangle, |q_3\rangle, |q_4\rangle, |t_1\rangle, |t_2\rangle, |d''\rangle$$

4. Lösche q_3 und q_4

$$|1\rangle, |a''\rangle, |b''\rangle |d''\rangle \xrightarrow{2\times\text{CRIGHTNEIGHBOUR}^\dagger} |1\rangle, |a'\rangle, |b'\rangle, |d'\rangle$$

$$|d'\rangle, |q_4\rangle \xrightarrow[\text{CNOT}]{d'\leq 0} |d\rangle, |0\rangle$$

$$|1\rangle, |a'\rangle, |b'\rangle |d'\rangle \xrightarrow{2\times\text{CRIGHTNEIGHBOUR}^\dagger} |1\rangle, |a\rangle, |b\rangle, |d\rangle$$

$$|d\rangle, |q_3\rangle \xrightarrow[\text{CNOT}]{d\leq 0} |d\rangle, |0\rangle$$

5. Gebe aus: $|a_1'\rangle$, $|b_1'\rangle$, $|a\rangle$, $|b\rangle$, $|d\rangle$, $|q_1'\rangle$, $|t_1' t_2'\rangle$.

5.4 Berechnung des Regulators

In diesem Abschnitt präsentieren wir einen Algorithmus zur Bestimmung des Regulators eines reell-quadratischen Zahlkörpers. Wir verwenden dazu wieder das Framework von der Seite 58. Die periodische Funktion, die wir dabei verwenden, bildet eine ganzen Zahl x auf das Ideal links von $x/4$ ab. Diese Funktion ist nicht periodisch im Sinne des Abschnitts 2.4.4. Insbesondere ist sie nicht mehr injektiv innerhalb einer Periode. Unten werden wir jedoch zeigen, daß die Berechnung des Regulators auch mit dieser Funktion möglich ist.

Eine ähnliche Funktion zur Berechnung des Regulators schlug Hallgren in [Hal02] vor. Hallgrens Funktion ordnete einer ganzen Zahl x das Paar $(\mathfrak{a}_-(x/N), N\lfloor \mathrm{dist}(\mathfrak{a}_-(x/N), x/N)\rfloor)$ zu, $N \in \mathbb{Z}$ mit $N > 2\sqrt{\Delta}$. Auf die Probleme mit der Genauigkeit der Logarithmenberechnung und dadurch auftretende Probleme mit der Bestimmung von $\mathfrak{a}_-(x)$ ging er nicht ein. Diese Probleme führen jedoch dazu, daß die Funktion entweder keine Polynomzeitfunktion ist oder, falls die Logarithmen approximiert werden, Lücken im Periodengitter aufweisen und zusätzlich die Injektivität innerhalb einer Periode verlieren kann. Unser Algorithmus, hingegen, behandelt die oben genannten Probleme und führt zu einem vollständigen Beweis, daß die Berechnung des Regulators in Quanten-Polynomzeit möglich ist. Darüberhinaus ist unser Algorithmus schneller und benötigt weniger Qubits als der Algorithmus von Hallgren.

Satz 5.4.1 *Sei Δ eine Diskriminante eines reell-quadratischen Zahlkörpers. Definiere* $\mathrm{Reg} : \mathbb{Z} \longrightarrow \mathcal{R}^+ : x \longmapsto \mathfrak{a}_-(x/4)$*, wobei hier die approximative Version von $\mathfrak{a}_-(x)$ verwendet wird.*

Die folgenden zwei Lemmas zeigen die Periodizität von Reg. Im Lemma 5.4.3 wird gezeigt, daß es in jedem Zyklus der Funktion Reg Bereiche ganzer Zahlen existieren, welche auf ein Ideal $f \in \mathcal{R}^+$ abgebildet werden, daß diese Bereiche zusammenhängend sind, daß die Größe dieser Bereiche höchstens $\ln\sqrt{\Delta} + 3$ ist und daß sich diese Größen von Zyklus zu Zyklus um höschstens fünf unterscheiden. Im Lemma 5.4.2 wird gezeigt, daß die jeweils ersten Zahlen in Bereichen verschiedener Zyklen mit Periode $\approx 4R$ vokommen.

Lemma 5.4.2 *Seien Δ eine Diskriminante eines reell-quadratischen Zahlkörpers $\mathcal{K}$ und R der Regulator von $\mathcal{K}$. Dann gilt*

$$\begin{aligned} \forall f \in \mathcal{R}^+.\exists y \in \mathbb{R}.\forall k \in \mathbb{Z}.\exists \epsilon \in \mathbb{R}, |\epsilon \leq 1|.y + 4kR + \epsilon \in \mathbb{Z} \text{ und} \\ \mathrm{Reg}(y + 4kR + \epsilon) = f \text{ und} \\ \mathrm{Reg}(y + 4kR + \epsilon - 1) = \rho^{-2}(f). \end{aligned} \tag{5.2}$$

Beweis. Seien $f \in \mathcal{R}^+$, $y = 4\,\mathrm{Log}\, f + 1/2$, $k \in \mathbb{Z}$ und $x \in \mathbb{Z}$, so daß $|x/4 - \mathrm{Log}\, f - kR| < 1/8$.

Wir rufen noch einmal in Erinnerung, daß die Funktion Reg eine approximative Logarithmusberechnung durchführt, die höchstens um $\pm 1/8$ vom tatsächlichen Logarithmus abweicht.

Satz 2.5.72 besagt, daß $\ln 2 < \mathrm{Log}\,\rho^2(f) - \mathrm{Log}\, f$ gilt. Daraus folgt, daß $\mathrm{Reg}(x+1) = f$ gilt, d.h. das Periodengitter enthält keine Lücken.

Wir unterscheiden zwei Fälle.

$\frac{x}{4} - \frac{1}{8} < \mathrm{Log}\, f + kR < \frac{x}{4}$**:** Es gilt entweder $\mathrm{Reg}(x) = f$ und $\mathrm{Reg}(x-1) = \rho^{-2}(f)$ oder $\mathrm{Reg}(x) = \rho^{-2}(f)$ und $\mathrm{Reg}(x+1) = f$. Im ersten Fall ist $|x - y - 4kR| < 1/2$. Im zweiten Fall gilt $|(x+1) - y - 4kR| < 1$.

$\frac{x}{4} < \operatorname{Log} f + kR < \frac{x}{4} + \frac{1}{8}$**:** Es gilt wieder entweder $\operatorname{Reg}(x) = f$ und $\operatorname{Reg}(x-1) = \rho^{-2}(f)$ oder $\operatorname{Reg}(x) = \rho^{-2}(f)$ und $\operatorname{Reg}(x+1) = f$. Im ersten Fall ist $|x - y - 4kR| < 1$. Im zweiten Fall gilt $|x - y - 4kR| < 1/2$.

Somit ist die Existenz von y und ϵ und damit das ganze Lemma bewiesen. □

Lemma 5.4.3 *Sei Δ die Diskriminante eines reell-quadratischen Zahlkörpers, dessen Regulator größer als $5 \ln \sqrt{\Delta}$ ist. Seien $f \in \mathcal{R}^+$, $y = 4 \operatorname{Log} f + 1/2$, $k \in \mathbb{Z}$ und $\epsilon_{(y,k)} \in \mathbb{R}$, so daß $y + 4kR + \epsilon_{(y,k)} \in \mathbb{Z}$, $\operatorname{Reg}(y + 4kR + \epsilon_{(y,k)}) = f$ und $\operatorname{Reg}(y + 4kR + \epsilon_{(y,k)} - 1) = \rho^{-2}(f)$. Dann existiert ein $m_{(y,k)} \in \mathbb{N}$, $0 \leq m_{(y,k)} < \ln \sqrt{\Delta} + 3$, so daß die folgenden Aussagen wahr sind:*

1. *Es gilt* $\operatorname{Reg}(y + 4kR + \epsilon_{(y,k)} + m_{(y,k)} + 1) = \rho^2(f)$.
2. *Für alle $m \in \mathbb{N}$, $0 \leq m \leq m_{(y,k)}$, gilt* $\operatorname{Reg}(y + 4kR + \epsilon_{(y,k)} + m) = f$.
3. *Es gilt* $\max_{k,k' \in \mathbb{Z}} |m_{(y,k)} - m_{(y,k')}| \leq 5$.

Beweis. Wir verwenden die Bezeichnungen aus dem Lemma.

Die erste Aussage folgt aus dem Satz 2.5.72, der besagt, daß $\operatorname{Log} \rho^2(f) - \operatorname{Log} f \leq \ln \sqrt{\Delta}$. Daraus folgt, daß $\operatorname{Reg}(y + 4kR + +\epsilon_{(y,k)}) \neq \operatorname{Reg}(y + 4kR + +\epsilon_{(y,k)} + \left\lceil \ln \sqrt{\Delta} \right\rceil + 2)$.

Die zweite und dritte Aussage folgt sofort aus der Definition von $\operatorname{Reg}(x) = f_{x/4}$ und dem der Approximationsgüte des Logarithmus. □

Satz 5.4.4 *Seien Δ eine Diskriminante eines reell-quadratischen Zahlkörpers, dessen Regulator größer als $12 \ln \sqrt{\Delta} + 48$ ist. Der Algorithmus* SampleDual-Reg*, welcher auf Seite 98 abgebildet ist, erhält als Eingabe:*

- *die Diskriminante Δ,*
- *eine Zweierpotenz q mit $q/2 \leq 5\Delta \ln^2 \Delta < q$,*
- *$\log_2 q$ Paare (f_i, d_i), $i = 1, \ldots, \log_2 q$, mit $\operatorname{Reg}(2^i) = f_i$ und $d_i \in \mathbb{Q}$, so daß $|\operatorname{Log} f_i - d_i + 2^{i-2}| < 1/(16 \log_2 q)$*

SampleDual-Reg *berechnet eine Approximation eines zufälligen Vektors aus dem Gitter $(q/R)\mathbb{Z}$. Die Approximation hat die Form $(q/R)z + \omega$ mit $z \in \mathbb{Z}$ und $|\omega| \leq 1/2$.*

Sei $n = \operatorname{size}(\Delta)$. Der Algorithmus benötigt für die internen Berechnungen höchstens $12,5n + 16 \log_2^2 n + 89 \log n + 150$ Qubits, falls die Quantenaddition verwendet wird, bzw. höchstens $13,5n + 16 \log_2^2 n + 93 \log n + 160$ Qubits, falls die klassische Addition verwendet wird.

Die Anzahl der elementaren Quantengatter ist $O(n^5)$, falls die klassische Addition verwendet wird, bzw. $O(n^6)$, falls die Quantenaddition verwendet wird.

Die Erfolgswahrscheinlichkeit des Algorithmus ist mindestens $1/2^{13}$.

Algorithm 39 SampleDual-Reg

Input: q, Δ, $(\mathfrak{a}_1, d_1)$, …, $(\mathfrak{a}_{\log_2 q}, d_{\log_2 q})$.
Output: Approximation einer Zahl aus $(q/R)\mathbb{Z}$.

1. Initialzustand
$$|0\rangle, |(1, \Delta \mod 2)\rangle.$$

2. Erzeuge Superposition
$$\xrightarrow{(\log_2 q)\times \mathrm{H}} \frac{1}{\sqrt{q}} \sum_{x=0}^{q-1} |x\rangle, |(1, \Delta \mod 2)\rangle.$$

3. Berechne die Funktion Reg.

   (a) Wende $(\log_2 q)$ mal die Funktion Multiplication-RQ mit q_i als Kontrollqubit, $\mathfrak{a}_1$, d_1, … $\mathfrak{a}_{\log_2 q}$, $d_{\log_2 q}$ als Eingabe
$$\xrightarrow{(\log_2 q)\times \text{Multiplication-RQ}} \frac{1}{\sqrt{q}} \sum_{x=0}^{q-1} |x\rangle, |\mathfrak{a}_-(x), d_x, \mathsf{temp}\rangle.$$

   (b) Kopiere
$$\xrightarrow{(\log_2 q)\times \text{Multiplication-RQ}} \frac{1}{\sqrt{q}} \sum_{x=0}^{q-1} |x\rangle, |\mathfrak{a}_-(x), d_x, \mathsf{temp}\rangle, |\operatorname{Reg}(x)\rangle.$$

   (c) Lösche temporäre Qubits
$$\xrightarrow{(\log_2 q)\times \text{Multiplication-RQ}^\dagger} \frac{1}{\sqrt{q}} \sum_{x=0}^{q-1} |x\rangle, |\operatorname{Reg}(x)\rangle.$$

4. Messe das zweite Register
$$\longrightarrow \frac{1}{\sqrt{p}} \sum_{k\in\mathcal{M}} \sum_{m=0}^{m_{(x',k)}} |x' + 4Rk + m + \epsilon_{(x',k)}\rangle, |\operatorname{Reg}(x')\rangle$$
mit einem zufälligen $x' \in \{0, \ldots, \lfloor 4R \rfloor\}$, $\epsilon_{(x',k)}$ und $m_{(x',k)}$ wie im Lemma 5.4.3, $\mathcal{M} = \mathcal{M}_{x'} = \{\, k \in \mathbb{Z} \mid 0 \le x' + 4Rk + \epsilon_{(x',k)} < q \,\}$ und $p = \operatorname{card} \{\, x \in \mathbb{Z} \mid 0 \le x < q \text{ und } \operatorname{Reg}(x) = \operatorname{Reg}(x') \,\}$.

5. Wende die Quantenfouriertransformation auf das erste Register an
$$\xrightarrow{\mathrm{QFT}} \frac{1}{2\sqrt{pq}} \sum_{y=0}^{4q-1} \sum_{k\in\mathcal{M}} \sum_{m=0}^{m_{(x',k)}} \exp\left(2\pi i \frac{x' + 4Rk + m + \epsilon_{(x',k)}}{4q} y\right) |y\rangle, |\operatorname{Reg}(x')\rangle.$$

6. Messe das erste Register und gebe das Ergebnis zurück.

Beweis. Wir verwenden die gleichen Bezeichnungen wie im Satz und im Algorithmus.

Als erstes analysieren wir die Erfolgswahrscheinlichkeit des Algorithmus, d.h. die Wahrscheinlichkeit, daß der Algorithmus eine Approximation der Form $z/(4R) + \omega$ mit $z \in \mathbb{Z}$ und $|\omega| \leq 1/(8q)$ zurückgibt. Dazu seien $m_{max} = \max_{k \in \mathcal{M}_{x'}} m_{(x',k)}$, $m_{min} = \min_{k \in \mathcal{M}_{x'}} m_{(x',k)}$ und

$$\mathcal{Y} = \{\, y \in \mathbb{Z} \mid 0 \leq y \leq \frac{q}{4(m_{max}+2)} \text{ und } \frac{y}{4q} = \frac{z}{4R} + \omega_y \text{ mit } z \in \mathbb{Z} \text{ und } |\omega_y| \leq \frac{1}{8q} \,\}. \quad (5.3)$$

Die Wahrscheinlichkeit ein $y \in \mathcal{Y}$ zu messen ist

$$\Pr(y \in \mathcal{Y}) = \frac{1}{4pq} \left| \sum_{k \in \mathcal{M}} \sum_{m=0}^{m_{(x',k)}} \exp\left(2\pi i \frac{4Rk + m + \epsilon_{(x',k)}}{4q} y\right) \right|^2 .$$

Wegen

$$\frac{(4Rk + m + \epsilon_{(x',k)})y}{4q} = \frac{4Rkl}{4R} + 4Rk\omega_{(x',k)} + \frac{(m + \epsilon_{(x',k)})y}{4q} \equiv 4Rk\omega_{(x',k)} + \frac{(m + \epsilon_{(x',k)})y}{4q} \mod 1$$

und der Periodizität der Funktion exp können wir schreiben

$$\Pr(y \in \mathcal{Y}) = \frac{1}{4pq} \left| \sum_{k \in \mathcal{M}} \sum_{m=0}^{m_{(x',k)}} \exp\left(2\pi i (4Rk\omega_y + \frac{(m + \epsilon_{(x',k)})y}{4q})\right) \right|^2 . \quad (5.4)$$

Aus Lemmas 5.4.2 und 5.4.3 und der Gleichung (5.3) folgt $|4Rk\omega_y| \leq 1/8$ und $-1/16 \leq (m + \epsilon_{(x',k)})y/(4q) \leq 1/16$. Das bedeutet, daß in (5.4) p Vektoren der Länge eins aufsummiert werden, die alle in einem Kreissegment mit Winkel $\pi/2$ liegen. Deshalb gilt für die Wahrscheinlichkeit, daß ein bestimmtes $y \in \mathcal{Y}$ gemessen wird

$$\Pr(y \in \mathcal{Y}) \geq \frac{1}{4pq} \left| p \frac{\sqrt{2}}{2} \right|^2 = \frac{p}{8q}.$$

Als nächstes schätzen wir p sowie die Mächtigkeit von $\mathcal{Y}$ von unten ab. Es gilt

$$p \geq (\operatorname{card} \mathcal{M}_{x'} - 1)(m_{min} + 1) + 1 \geq \left(\frac{q}{4R} - \frac{9}{4}\right)(m_{min} + 1) + 1 \geq \frac{q}{8R}(m_{min} + 1)$$

$$\operatorname{card} \mathcal{Y} \geq \{\, z \in \mathbb{Z} \mid 1 \leq z \leq \frac{R}{4(m_{max}+1)} - \frac{R}{2q} \,\} \geq \frac{R}{4(m_{max}+1)} - \frac{3}{2} \geq \frac{R}{8(m_{max}+1)}.$$

Daraus folgt, daß

$$\sum_{y \in \mathcal{Y}} \Pr(y \in \mathcal{Y}) \geq \frac{p}{8q} \operatorname{card} \mathcal{Y} \geq \frac{1}{64R} \frac{(m_{min}+1)R}{8(m_{max}+2)} \geq \frac{1}{2^{13}}$$

ist.

Die Anzahl der Qubits und Gatter folgt aus dem Satz 5.3.1. Um die gewünschte Approximation der Abstände zu erreichen, reicht es $k = \lfloor \log_2 n \rfloor + 5$ und $p = \lfloor 4 \log_2 n + 10 \rfloor$ zu wählen. Falls die Quantenaddition verwendet wird, werden höchstens

$$n+p+k+2+7, 5n+\log_2 n+25+\max\{\frac{n}{2}, p^2+\log_2 p\}+n+3\log_2 q \leq 12, 5n+16\log_2^2 n+89\log n+150$$

Qubits benötigt. Falls die klassische Addition verwendet wird, werden höchstens

$$13,5n + 16\log_2^2 n + 93\log n + 160$$

Qubits benötigt.

Die Anzahl der elementaren Quantengatter ist $O(n^5)$, falls die klassische Addition verwendet wird, bzw. $O(n^6)$, falls die Quantenaddition verwendet wird. □

Als nächstes präsentieren wir den Algorithmus zur Berechnung des Regulators.

Algorithm 40 REGULATOR

Input: Die Diskriminante Δ eines reell-quadratischen Zahlkörpers $\mathcal{K}$.
Output: Den Regulator R von $\mathcal{K}$.

1. Teste klassisch, ob $R < 2^{10}\ln\Delta$ gilt. Falls ja, berechne die geforderte Approximation von R und gehe zu Schritt 7.
2. Setze $q = 2^x$, so daß $q/2 < 5\Delta\ln^2\Delta \leq q$.
3. Berechne klassisch $\mathfrak{a}_i = \mathrm{Reg}(2^i)$ und $d_i \in \mathbb{Q}$ für $i = 1,\ldots,\log_2 q$, wobei $|\mathrm{dist}(\mathcal{O}_\Delta, \mathfrak{a}_i) - d_i - \lfloor 2^{i-2}/R \rfloor R + 2^{i-2}| \leq 1/(16\log_2 q)$ ist.
4. Berechne mit SAMPLEDUAL-REG $y_1 = (q/R)z_1 + \omega_1$ und $y_2 = (q/R)z_2 + \omega_2$, $|\omega_1|, |\omega_2| \leq 1/2$, welche die Vektoren in $(q/R)\mathbb{Z}$ approximieren.
5. O.B.d.A. sei $y_1 \leq y_2$. Mit dem Algorithmus zur Kettenbruchentwicklung angewandt auf y_1/y_2 berechne z_1 und z_2. Dann ist qz_1/y_2 eine Approximation des Regulators.
6. Berechne die geforderte Approximation von R mit klassischen Algorithmen (siehe [BB94], [Thi95] oder [Mau00]).
7. Gebe die gefundene Approximation von R aus.

Lemma 5.4.5 *Sei $q \in \mathbb{Z}$ wie in* REGULATOR. *Seien außerdem $y_i = (q/R)z_i + \omega_i$ mit $z_i \in \mathbb{Z}$ und $i = 1,2$ zwei Zahlen die im Schritt 3 von* REGULATOR *gemessen wurden. Ist $y_1 < y_2$, dann gilt*

$$\left|\frac{y_1}{y_2} - \frac{z_1}{z_2}\right| \leq \frac{1}{2z_2^2}.$$

Beweis. Es gilt

$$\left|\frac{y_1}{y_2} - \frac{z_1}{z_2}\right| \leq \left|\frac{qz_1 + R\omega_1}{qz_2 + R\omega_2} - \frac{z_1}{z_2}\right| = \left|\frac{qz_1z_2 + z_2R\omega_1 - qz_1z_2 - z_1R\omega_2}{z_2(qz_2 + R\omega_2)}\right|$$
$$\leq \frac{R}{2}\left|\frac{z_1 + z_2}{z_2(qz_2 + R\omega_2)}\right| \leq \frac{R}{qz_2 - R/2} \leq \frac{1}{2z_2^2}.$$

Die letzte Ungleichung gilt wegen der Wahl $q > R^2$ und $y \in \mathcal{Y}$ mit $\mathcal{Y}$ aus (5.3).

Satz 5.4.6 *Seien Δ eine Diskriminante eines reell-quadratischen Zahlkörpers $\mathcal{K}$. Bei Eingabe von Δ berechnet der Algorithmus* Regulator *den Regulator von $\mathcal{K}$.*

Sei $n = \text{size}(\Delta)$. Der Algorithmus benötigt für die internen Berechnungen höchstens $12,5n + 16\log_2^2 n + 89\log n + 150$ Qubits, falls die Quantenaddition verwendet wird, bzw. höchstens $13,5n + 16\log_2^2 n + 93\log n + 160$ Qubits, falls die klassische Addition verwendet wird.

Die Anzahl der elementaren Quantengatter ist $O(n^5)$, falls die klassische Addition verwendet wird, bzw. $O(n^6)$, falls die Quantenaddition verwendet wird.

Die Komplexität der klassischen Vor- und Nachberechnungen ist polynomiell.

Die Erfolgswahrscheinlichkeit des Algorithmus beträgt mindestens 2^{-28}.

Beweis. Wir verwenden die gleichen Bezeichnungen wie im Satz und im Algorithmus.

Sei zuerst $R < 2^{10}\ln\Delta$. Dann läßt sich der Regulator klassisch in Polynomzeit berechnen (siehe [BB94]). Sei nun $R > 2^{10}\ln\Delta$, damit ist sichergestellt, daß die Menge $\mathcal{Y}$ in (5.3) mindestens eine Approximation einer Zahl aus $(1/R)\mathbb{Z}$ enthält. Die Wahl von q stellt sicher, daß die Vorbedingungen in Lemma 5.4.5 erfüllt sind, so daß im Schritt 5 des Algorithmus die Kettenbruchentwicklung die Zahl z_1 und z_2 berechnet (unter der Voraussetzung, daß $\gcd(z_1, z_2) = 1$ gilt). Die Zahl qz_1/y_2 ist eine Approximation des Regulators. Diese Approximation läßt sich mit klassischen Algorithmen (siehe [BB94], [Thi95] oder [Mau00]) zu einer gewünschter Präzision verbessern.

Wir bestimmen nun die untere Schranke für die Erfolgswahrscheinlichkeit des Algorithmus. Sie folgt aus der Wahrscheinlichkeit, daß im Schritt 4 des Algorithmus $y_1, y_2 \in \mathcal{Y}$ gemessen werden und der Wahrscheinlichkeit, daß $\gcd(z_1, z_2) = 1$ gilt. Aus Lemma 2.2.3 und Satz 5.4.4 folgt, daß die Erfolgswahrscheinlichkeit mindestens

$$\frac{1}{4}\frac{1}{2^{13}}\frac{1}{2^{13}} \geq 2^{-28}$$

beträgt.

Die Anzahl der benötigten Qubits und elementaren Quantengatter folgt direkt aus dem Satz 5.4.4. □

5.5 Lösung des Hauptidealproblems

In diesem Abschnitt präsentieren wir einen Algorithmus zur Lösung des folgenden Problems:

Problem 5.5.1 (Hauptidealproblem) *Gegeben ein Hauptideal $\mathfrak{a}$, finde eine Approximation der Distanz $S \in \mathbb{R}$ zwischen $\mathcal{O}$ und $\mathfrak{a}$.*

Um das Hauptidealproblem zu lösen, erweitern wir die Funktion Reg aus dem letzten Abschnitt zur folgenden Funktion:

Satz 5.5.2 *Seien $\Delta > 0$ eine Diskriminante und $\mathfrak{b}$ ein $\mathcal{O}_\Delta$-Ideal. Wähle einen Algorithmus zur Berechnung der approximativen Abstände zwischen zwei Idealen. Die Abbildung* Ord_{RQ} *ist wie folgt definiert:*

$$\mathrm{Ord}_{RQ} : \mathbb{Z} \times \mathbb{Z} \longrightarrow \mathcal{R}^+ \times \mathbb{Z} : (x, y) \longmapsto (\mathfrak{a}_{(x,y)}, z).$$

Dabei ist $\mathfrak{a}_{(x,y)}$ ein Ideal mit folgenden Eigenschaften: Sei d der mit dem gewählten Algorithmus berechnete Abstand zwischen $\mathfrak{b}^x$ und $\mathfrak{a}_{(x,y)}$ und e der mit dem gewählten Algorithmus berechnete Abstand zwischen $\mathfrak{a}_{(x,y)}$ und $\rho(\rho(\mathfrak{a}_{(x,y)}))$, dann gilt $d \leq y/4$ und $d + e \geq y/4$. Für die Zahl z gilt $z = \lfloor y/4 - d \rfloor$.

Analog zu Lemma 5.4.2 und 5.4.3 läßt sich das folgende Lemma zeigen.

Lemma 5.5.3 *Seien Δ eine Diskriminante eines reell-quadratischen Zahlkörpers mit Regulator R und $\mathfrak{b}$ ein $\mathcal{O}_\Delta$-Ideal, so daß $\mathrm{dist}(\mathcal{O}_\Delta, \mathfrak{b}^n) = S$ gilt, wobei n die kleinste Zahl größer als Null mit $\mathcal{O}_\Delta \sim \mathfrak{b}^n$ ist. Sei weiterhin L ein Gitter mit der Basis $((n, -4S)^t, (0, R)^t)$. Dann existieren für mindestens die Hälfte aller Elemente $(\mathfrak{a}, z)$ aus $\{ \mathrm{Ord}_{RQ}(x, y) \mid (x, y) \in \mathbb{Z}^2 \}$ Zahlen $x_1' \in \mathbb{Z}$ und $x_2' \in \mathbb{R}$, so daß es für alle $(x_1, x_2) \in \mathbb{Z}^2$ ein $\epsilon_{(\mathbf{x},x_2')}$, $|\epsilon_{(\mathbf{x},x_2')}| < 1$, gibt, so daß gilt:*

$$\mathrm{Ord}_{RQ}(x_1' + x_1 n, x_2' - 4x_1 S + 4x_2 R + m + \epsilon_{(\mathbf{x},x_2')}) = (\mathfrak{a}, z)$$

mit $0 \leq m < m_{(\mathbf{x},x_2')} \leq 6$. □

Das Gitter L nennen wir das Periodengitter von Ord_{RQ}.

Satz 5.5.4 *Seien Δ eine Diskriminante eines reell-quadratischen Zahlkörpers. Außerdem sei*

$$\begin{aligned}\mathcal{Y} = \{ (y_1, y_2) \in \mathbb{Z}^2 \mid 0 \leq y_1 < 8q \text{ und } \frac{y_1}{8q} = \frac{z_1}{n} + \frac{z_2 S}{nR} + \omega_1 \text{ mit } z_1, z_2 \in \mathbb{Z} \text{ und } |\omega_1| \leq \frac{1}{16q} \\ 0 \leq y_2 < \frac{q}{m_{max} + 2} \text{ und } \frac{y_2}{8q} = \frac{z_2}{4R} + \omega_2 \text{ mit } |\omega_2| \leq \frac{1}{16q} \}.\end{aligned} \tag{5.5}$$

Mit einer Wahrscheinlichkeit größer 2^{-17} berechnet SampleDual-Reg *Vektoren $(y_1/(8q), y_2/(8q))$ mit $(y_1, y_2) \in \mathcal{Y}$.*

Sei $n = \mathrm{size}(\Delta)$. Der Algorithmus benötigt für die internen Berechnungen höchstens $13,5n + 16\log_2^2 n + 92 \log n + 150$ Qubits, falls die Quantenaddition verwendet wird, bzw. höchstens $15,5n + 16\log_2^2 n + 96 \log n + 160$ Qubits, falls die klassische Addition verwendet wird.

Die Anzahl der elementaren Quantengatter ist $O(n^5)$, falls die klassische Addition verwendet wird, bzw. $O(n^6)$, falls die Quantenaddition verwendet wird.

Beweis. Wir verwenden die Bezeichnungen aus dem Algorithmus und dem Satz.

Als erstes schätzen wir die Erfolgswahrscheinlichkeit des Algorithmus.

Die Wahrscheinlichkeit ein $\mathbf{y} \in \mathcal{Y}$ zu messen, ist

$$\Pr(\mathbf{y} \in \mathcal{Y}) = \frac{1}{64q^2 p} \left| \sum_{x_1=0}^{\lfloor (q - x_1 - 1)/n \rfloor} \sum_{x_2 \in \mathcal{M}} \sum_{m=0}^{m_{(\mathbf{x},x_2')}} e^{\frac{2\pi i}{8q} \left(y_1 x_1 n + (-4x_1 S + 4x_2 R + m + \epsilon_{(\mathbf{x},x_2')}) y_2 \right)} \right|^2 \tag{5.6}$$

Algorithm 41 SampleDual-Ord-RQ

Input: q, Δ, $(\mathfrak{a}_1, d_1)$, ... $(\mathfrak{a}_{2\log_2 q}, d_{2\log_2 q})$.
Output: Approximation eines Vektoren aus dem zu $((n, -4S)^t, (0, R)^t)$ dualen Gitter.

1. Initialzustand
$$|0\rangle|0\rangle|(1, \Delta \bmod 2)\rangle.$$

2. Erzeuge Superposition
$$\overset{2\log_2 q \times \mathrm{H}}{\longrightarrow} \frac{1}{q} \sum_{x_1=0}^{q-1} \sum_{x_2=0}^{q-1} |x_1\rangle|x_2\rangle|(1, \Delta \bmod 2)\rangle.$$

3. Berechne die Funktion Ord_{RQ}. Dazu verwende $(2\log_2 q)$ mal die Funktion Multiplication-RQ mit $(\mathfrak{a}_1, d_1)$, ... $(\mathfrak{a}_{2\log_2 q}, d_{2\log_2 q})$ als Eingabe
$$\overset{2\log_2 q \times \text{Multiplication-RQ}}{\longrightarrow} \frac{1}{q} \sum_{x_1=0}^{q-1} \sum_{x_2=0}^{q-1} |x_1\rangle|x_2\rangle|\,\mathrm{Ord}_{RQ}(x_1, x_2)\rangle.$$

4. Messe das dritte Register
$$\frac{1}{\sqrt{p}} \sum_{x_1=0}^{\lfloor (q-x_1-1)/n \rfloor} \sum_{x_2 \in \mathcal{M}} \sum_{m=0}^{m_{(\mathbf{x}, x_2')}} |x_1' + x_1 n\rangle|x_2' - 4x_1 S + 4x_2 R + m + \epsilon_{(\mathbf{x}, x_2')}\rangle|\,\mathrm{Ord}_{RQ}(\mathbf{x}')\rangle,$$
mit $x_1' \in \{0, \ldots, n-1\}$ und $x_2' \in \{0, \ldots, \lfloor 4R \rfloor\}$, mit $m_{(\mathbf{x}, x_2')}$ und $\epsilon_{(\mathbf{x}, x_2')}$ wie im Lemma 5.5.3, mit $\mathcal{M} = \mathcal{M}_{x_1, \mathbf{x}'} = \{\, x_2 \in \mathbb{Z} \mid 0 \le x_2' - 4x_1 S + 4x_2 R + \epsilon_{(\mathbf{x}, x_2')} < q \,\}$ und mit $p = (\lfloor (q - x_1 - 1)/n \rfloor + 1) \sum_{x_2 \in \mathcal{M}} (m_{(\mathbf{x}, x_2')} + 1)$.

5. Wende die Quantenfouriertransformation auf die ersten zwei Register an
$$\frac{1}{8q\sqrt{p}} \sum_{y_1, y_2=0}^{8q-1} \sum_{x_1=0}^{\lfloor (q-x_1-1)/n \rfloor} \sum_{x_2 \in \mathcal{M}} \sum_{m=0}^{m_{(\mathbf{x}, x_2')}} \exp\left(2\pi i \frac{x_1' + x_1 n}{8q} y_1\right) |y_1\rangle \times$$
$$\times \exp\left(\frac{x_2' - 4x_1 S + 4x_2 R + m + \epsilon_{(\mathbf{x}, x_2')}}{8q} y_2\right) |y_2\rangle|\,\mathrm{Ord}_{RQ}(\mathbf{x}')\rangle.$$

6. Messe die ersten beiden Register (y_1, y_2) und gebe $(y_1/(8q), y_2/(8q))$ zurück.

Wegen

$$x_1 n\frac{y_2}{8q} + (-4x_1S + 4x_2R + m + \epsilon_{(\mathbf{x},x_2')})\frac{y_2}{8q} =$$
$$x_1 n\left(\frac{z_1}{n} + \frac{z_2 S}{nR} + \omega_1\right) + (-4x_1S + 4x_2R + m + \epsilon_{(\mathbf{x},x_2')})\left(\frac{z_2}{4R} + \omega_2\right) \equiv$$
$$x_1 n\omega_1 + (-4x_1S + 4x_2R)\omega_2 + (m + \epsilon_{(\mathbf{x},x_2')})\frac{y_2}{8q} \pmod 1$$

mit $|\omega_1|, |\omega_2| \le 1/(16q)$ und $0 \le y_2 < q/(m_{max}+2)$ besteht die Summe in (5.6) aus p Vektoren der Länge Eins, die alle in einem Kreissegment mit Winkel $\pi/2$ liegen. Daraus folgt

$$\Pr(\mathbf{y} \in \mathcal{Y}) \ge \frac{p}{128q^2}$$

Als nächstes schätzen wir p und $\operatorname{card}\mathcal{Y}$ von unten ab. Es gilt

$$p = (\lfloor (q - x_1 - 1)/n \rfloor + 1) \sum_{x_2 \in \mathcal{M}} (m_{(\mathbf{x},x_2')} + 1) \ge \frac{q}{n}\frac{q}{8R}(m_{min} + 1)$$

und

$$\begin{aligned}
\operatorname{card}\mathcal{Y} = \operatorname{card}\{\, (z_1, z_2) \in \mathbb{Z}^2 \mid\ & 0 \le \frac{z_1}{n} + \frac{z_2 S}{nR} + \omega_1 < 1 \text{ und} \\
& 0 \le \frac{z_2}{4R} + \omega_2 \le \frac{1}{8(m_{max}+2)},\ \text{mit } |\omega_1|, |\omega_2| \le \frac{1}{16q}\,\} \\
\ge \operatorname{card}\{\, (z_1, z_2) \in \mathbb{Z}^2 \mid\ & \frac{1}{16q} \le \frac{z_1}{n} + \frac{z_2 S}{nR} < \frac{16q-1}{16q} \text{ und } \frac{1}{16q} \le \frac{z_2}{4R} \le \frac{1}{8(m_{max}+2)} - \frac{1}{16}\,\} \\
\ge\ & \frac{nR}{8(m_{max}+2)}.
\end{aligned}$$

Wir fassen zusammen und erhalten

$$\sum_{y \in \mathcal{Y}} \Pr(y \in \mathcal{Y}) \ge \frac{nR}{8(m_{max}+2)}\frac{q}{n}\frac{q}{8R}(m_{min}+1)\frac{1}{128q^2} \ge \frac{1}{2^{16}}.$$

Wegen Lemma 5.5.3 ist die Erfolgswahrscheinlichkeit des Algorithmus mindestens 2^{-17}.

Für die Berechnung der Anzahl der Qubits verwenden wir den Satz 5.4.4. Zusätzlich zu den im Satz 5.4.4 ermittelten Qubits kommen Qubits für die Darstellung von x_2 und der Zahl z im Paar $(\mathfrak{a}, z) = \operatorname{Ord}_{RQ}(x_1, x_2)$ hinzu.

Die Anzahl der elementaren Quantengatter ist die gleiche wie im Satz 5.4.4. □

Satz 5.5.5 *Seien Δ die Diskriminante eines reell-quadratischen Zahlkörpers und $\mathfrak{a}$ ein $\mathcal{O}_\Delta$-Hauptideal. Bei der Eingabe von Δ und $\mathfrak{a}$ berechnet* HAUPTIDEAL *eine Approximation des Abstands* $\operatorname{dist}(\mathcal{O}, \mathfrak{a})$. *Die Erfolgswahrscheinlichkeit des Algorithmus ist mindestens $1/2^{65}$.*

Sei $n = \operatorname{size}(\Delta)$. Der Algorithmus benötigt für die internen Berechnungen höchstens $13,5n + 16\log_2^2 n + 92\log n + 150$ Qubits, falls die Quantenaddition verwendet wird, bzw. höchstens $15,5n + 16\log_2^2 n + 96\log n + 160$ Qubits, falls die klassische Addition verwendet wird.

Die Anzahl der elementaren Quantengatter ist $O(n^5)$, falls die klassische Addition verwendet wird, bzw. $O(n^6)$, falls die Quantenaddition verwendet wird.

Die klassischen Vor- und Nachberechnungen können in Polynomzeit durchgeführt werden.

Algorithm 42 HAUPTIDEAL

Input: Die Diskriminante Δ eines reell-quadratischen Zahlkörpers und ein Hauptideal $\mathfrak{a}$.
Output: Eine Approximation des Abstands $S = \mathrm{dist}(\mathcal{O}, \mathfrak{a})$.

1. Prüfe klassisch, ob $R < 64 \ln \Delta$. Falls ja gehe zu 9.
2. Berechne den Regulator R mit REGULATOR.
3. Setze $q = 2^x$, so daß $2q < \Delta \ln^2 \Delta \leq 4q$.
4. Berechne klassisch $f_i = \mathrm{Reg}(2^i)$ und $d_i \in \mathbb{Q}$ für $i = 1, \ldots, \log_2 q$, wobei $|\mathrm{dist}(\mathcal{O}_\Delta, f_i) - d_i - \left\lfloor 2^{i-2}/R \right\rfloor R + 2^{i-2}| \leq 1/(32 \log_2 q)$ ist.
5. Berechne klassisch f_i' und d_i' für $i = 1, \ldots, \log_2 q$, so daß gilt: $f_i' = (a, b)$ ist das erste reduzierte Ideal mit $a > 0$ in der Reduktionssequenz von $\mathfrak{a}^i$ und $d_i' \in \mathbb{Q}$ mit $|\mathrm{dist}(\mathfrak{a}i, f_i') - d_i'| \leq 1/(32 \log_2 q)$.
6. Führe zweimal SAMPLEDUAL-RQ aus. Seien (y_1, y_2) und (y_1', y_2') die zurückgegebenen Paare.
7. Berechne $z_2 = \lfloor y_2 R/(2q) \rceil$ und $z_2' = \lfloor y_2' R/(2q) \rceil$. Berechne zwei Zahlen k_1 und k_2, so daß $k_1 z_2 + k_2 z_2' = \gcd(z_2, z_2')$.
8. Berechne $p = y_1 k_1 + y_1' k_2 \mod 8q$ und $S' = pR/8q$. Falls $\gcd(z_2, z_2') = 1$ gilt, dann ist S' eine Approximation des gesuchten Abstands.
9. Bestimme den Abstand S mit der geforderten Approximationsgüte mit klassischen Algorithmen (siehe [BB94], [Thi95] oder [Mau00]).
10. Gebe die berechnete Approximation von S aus.

Beweis. Wir verwenden die gleichen Bezeichnungen wie im Satz und dem Algorithmus.

Im Schritt 1 testen wir, ob $R < 64 \ln \Delta$ gilt. Ist das der Fall, dann kann die gesuchte Approximation von S in klassischer Polynomzeit mit Algorithmen (siehe [BB94], [Thi95] oder [Mau00]) bestimmt werden. Wir nehmen nun an, daß $S \geq 64 \ln \Delta$ gilt.

Als erstes zeigen wir, daß HAUPTIDEAL eine Approximation von S zurückgibt, wenn $(y_1, y_2) \in \mathcal{Y}$, $(y_1', y_2') \in \mathcal{Y}$ und $\gcd(z_2, z_2') = 1$ gilt. Da $S \geq 64 \ln \Delta$ gilt, enthält die Menge $\mathcal{Y}$ aus (5.5) mindestens einen Vektor ungleich $(0, 0)$. Wegen (5.5) gilt

$$\frac{y_2}{8q} = \frac{z_2}{4R} + \omega_2, \quad |\omega_2| \leq \frac{1}{16q},$$

daraus folgt

$$z_2 = \frac{y_2 R}{2q} - 4R\omega_2 = \frac{y_2 R}{2q} + \omega' = \left\lfloor \frac{y_2 R}{2q} \right\rceil, \quad |\omega'| \leq \frac{1}{4}.$$

Analog läßt sich der Wert von z_2' bestimmen. Aus $k_1 z_2 + k_2 z_2' = \gcd(z_2, z_2') = 1$ folgt

$$\frac{y_1 k_1 + y_1' k_2}{8q} = k_1 z_1 + k_2 z_1' + \frac{S}{R} + \omega, \quad |\omega| \leq \frac{k_1 + k_2}{16q} < \frac{1}{R}.$$

Da $k_1 z_1 + k_2 z_1' \in \mathbb{Z}$ und $0 \leq S/R < 1$ gilt, ist der Wert $S' = pR/(8q)$ mit $p = y_1 k_1 + y_1' k_2 \mod 8q$ eine Approximation von S. Diese Approximation läßt sich mit klassischen Algorithmen (siehe [Mau00]) bis auf die gesuchte Approximationsgüte verbessern.

Als nächstes schätzen wir die Erfolgswahrscheinlichkeit des Algorithmus ab. Sie hängt davon ab, daß im ersten Schritt der Regulator richtig berechnet wird, daß im Schritt 6 zwei Vektoren $(y_1, y_2), (y_1', y_2') \in \mathcal{Y}$ mit $\mathcal{Y}$ aus (5.5) gemessen werden und daß $\gcd(z_2, z_2') = 1$ gilt. Aus Lemma 5.5.3 und Sätzen 5.4.6, 5.5.4 und 2.2.3 folgt, daß die Erfolgswahrscheinlichkeit von HAUPTIDEAL mindestens

$$\frac{1}{2^{28}} \frac{1}{2^{17}} \frac{1}{2^{17}} \frac{1}{4} \frac{1}{2} \geq \frac{1}{2^{65}}$$

beträgt.

Die Anzahl der Qubits und elementaren Quantengatter folgt sofort aus dem Satz 5.4.6 und dem Satz 5.5.4.

Aus [Mau00] folgt, daß alle klassischen Berechnungen in Polynomzeit durchgeführt werden. □

An dieser Stelle wollen wir noch bemerken, daß die Berechnung des Regulators im Schritt 2 vom restlichen Algorithmus entkoppelt werden kann. Deshalb ist die erwartete Anzahl der Wiederholungen für REGULATOR 2^{28} und die erwartete Anzahl der Wiederholungen für die Schritte 6 bis 8 2^{37}. Das ist wesentlich besser als die 2^{65}.

5.6 Berechnung der Ordnung

Wir erweitern den Algorithmus aus dem letzten Abschnitt, um auch die Ordnung eines Ideals in der reell-quadratischen Klassengruppe zu bestimmen.

Algorithm 43 ORD-RQ

Input: Die Diskriminante Δ eines reell-quadratischen Zahlkörpers und ein $\mathcal{O}_\Delta$-Ideal $\mathfrak{a}$.
Output: Die Ordnung n und eine Approximation des Abstands $\mathrm{dist}(\mathcal{O}, \mathfrak{a}^n)$.

1. Berechne den Regulator R mit REGULATOR.
2. Setze $q = 2^x$, so daß $q/2 < 2^{29}\Delta\sqrt{\Delta}\ln^3\Delta \leq q$.
3. Berechne klassisch $f_i = \mathrm{Reg}(2^i)$ und $d_i \in \mathbb{Q}$ für $i = 1, \ldots, \log_2 q$, wobei $|\mathrm{dist}(\mathcal{O}_\Delta, f_i) - d_i - \left\lfloor 2^{i-2}/R \right\rceil R + 2^{i-2}| \leq 1/(16\log_2 q)$ ist.
4. Berechne klassisch f_i' und d_i' für $i = 1, \ldots, \log_2 q$, so daß gilt: $f_i' = (a, b)$ ist das erste reduzierte Ideal mit $a > 0$ in der Reduktionssequenz von $\mathfrak{a}^i$ und $d_i' \in \mathbb{Q}$ mit $|\mathrm{dist}(\mathfrak{a}^i, f_i') - d_i'| \leq 1/(16\log_2 q)$.
5. Führe viermal SAMPLEDUAL-RQ aus. Seien $\mathbf{y}_i \in \mathbb{Z}^2$, $i = 1, 2, 3, 4$ die zurückgegebenen Paare.
6. Mit Algorithmus aus Satz 2.4.24 berechne eine Basis des Periodengitters der Funktion Ord_{RQ}.
7. Bestimme den Abstand S mit der geforderten Approximationsgüte mit klassischen Algorithmen (siehe [BB94], [Thi95] oder [Mau00]).
8. Gebe n und S aus.

Satz 5.6.1 *Seien Δ die Diskriminante eines reell-quadratischen Zahlkörpers und $\mathfrak{a}$ ein $\mathcal{O}_\Delta$-Ideal. Bei der Eingabe von Δ und $\mathfrak{a}$ berechnet* ORD-RQ *eine Basis des Periodengitters von* Ord_{RQ} *Die Erfolgswahrscheinlichkeit des Algorithmus ist mindestens* 2^{-106}.

Sei $n = \mathrm{size}(\Delta)$. Der Algorithmus benötigt für die internen Berechnungen höchstens $14n + 16\log_2^2 n + 93\log n + 204$ Qubits, falls die Quantenaddition verwendet wird, bzw. höchstens $16n + 16\log_2^2 n + 97\log n + 214$ Qubits, falls die klassische Addition verwendet wird.

Die Anzahl der elementaren Quantengatter ist $O(n^6)$, falls die klassische Addition verwendet wird, bzw. $O(n^7)$, falls die Quantenaddition verwendet wird.

Die Komplexität der klassischen Vor- und Nachberechnungen ist polynomiell.

Beweis. Wir verwenden die gleichen Bezeichnungen wie im Satz und im Algorithmus.

Die Wahl von q ist notwendig, um die Bedingungen des Satzes 2.4.24 zu erfüllen. Mit diesem q berechnet der Algorithmus aus Satz 2.4.24 das Periodengitter von Ord_{RQ} aus den Approximationen des Erzeugendensystem des dualen Gitters.

Als nächstes schätzen wir die Erfolgswahrscheinlichkeit von ORD-RQ ab. Sie hängt von der Wahrscheinlichkeit ab, mit der die Vektoren $\mathbf{y}_i \in \mathcal{Y}$ mit $\mathcal{Y}$ aus (5.5) im Schritt 5 gemessen werden und mit der diese Vektoren das ganze Gitter erzeugen, welches zum Periodengitter von Ord_{RQ} dual ist. Aus Sätzen 5.4.6, 5.5.4 und 2.4.23 folgt, daß die Erfolgswahrscheinlichkeit

des Algorithmus mindestens

$$\frac{1}{2^{28}}\left(\frac{1}{2^{17}}\right)^4\frac{1}{2^{10}} \geq 2^{-106}$$

beträgt.

Die Anzahl der elementaren Quantengatter folgt aus dem Satz 5.3.1 und der Tatsache, daß $q < 2^{30}\Delta\sqrt{\Delta}\ln^3\Delta$ gilt. □

5.7 Berechnung des diskreten Logarithmus

In diesem Abschnitt entwickeln wir einen Algorithmus zum Lösen des folgenden Problems

Problem 5.7.1 (Das erweiterte DL-Problem) *Seien $\Delta > 0$ eine Diskriminante und $\mathfrak{a}$ und $\mathfrak{b}$ zwei $\mathcal{O}_\Delta$-Ideale. Gegeben $\mathfrak{a}$ und $\mathfrak{b}$, finde, falls existiert, das minimale $n \in \mathbb{N}$ und ein $S \in \mathbb{Q}$, so daß es ein $\gamma \in \mathcal{K}$ mit $\mathfrak{a}^n = \gamma\mathfrak{b}$ gibt und $|S - \operatorname{Log}\gamma| < 1$ gilt.*

Satz 5.7.2 *Seien $\Delta > 0$ eine Diskriminante, $\mathfrak{a}$ und $\mathfrak{b}$ reduzierte $\mathcal{O}_\Delta$-Ideale. Wähle einen Algorithmus zur Berechnung der approximativen Abstände zwischen zwei Idealen. Die Abbildung DL_{RQ} ist wie folgt definiert:*

$$\mathrm{DL}_{RQ} : \mathbb{Z}^3 \longrightarrow \mathcal{R}^+ \times \mathbb{Z} : (x,y,z) \longmapsto (\mathfrak{a}_{(x,y,z)}, t).$$

Dabei ist $\mathfrak{a}_{(x,y,t)}$ ein Ideal mit folgenden Eigenschaften: Sei d der mit dem gewählten Algorithmus berechnete Abstand zwischen $\mathfrak{a}^x\mathfrak{b}^y$ und $\mathfrak{a}_{(x,y,z)}$ und e der mit dem gewählten Algorithmus berechnete Abstand zwischen $\mathfrak{a}_{(x,y,z)}$ und $\rho(\rho(\mathfrak{a}_{(x,y,z)}))$, dann gilt $d \leq z/4$ und $d + e \geq z/4$. Für die Zahl t gilt $t = \lfloor (y/4 - d) \rfloor$.

Lemma 5.7.3 *Seien Δ eine Diskriminante eines reell-quadratischen Zahlkörpers mit Regulator R und $\mathfrak{a}$ und $\mathfrak{b}$ zwei $\mathcal{O}_\Delta$-Ideale, so daß $\operatorname{dist}(\mathfrak{a}^n, \mathfrak{b}) = S$ gilt, wobei n die kleinste Zahl größer als Null mit $\mathfrak{a}^n \sim \mathfrak{b}$ ist. Sei weiterhin $L \subset \mathbb{R}$ ein Gitter mit der Basis $((n,-1,-4S)^t, (k,0,-4T)^t, (0,0,R)^t)$. Dann existieren für mindestens die Hälfte aller Elemente $(\mathfrak{a}, z)$ aus $\{\, \mathrm{DL}_{RQ}(\mathbf{x}) \mid \mathbf{x} \in \mathbb{Z}^3 \,\}$ Vektoren $\mathbf{x}' \in \mathbb{Z}^2 \times \mathbb{R}$, so daß es für alle $\mathbf{x} \in L$ ein $\epsilon_{(\mathbf{x},x_3')}$, $|\epsilon_{(\mathbf{x},x_3')}| < 1$, gibt, so daß gilt:*

$$\mathrm{DL}_{RQ}(\mathbf{x}' + \mathbf{x} + (0,0,m + \epsilon_{(\mathbf{x},x_2')})) = (\mathfrak{a}, z)$$

mit $0 \leq m < m_{(\mathbf{x},x_3')} \leq 6$. □

Das Gitter L aus dem Lemma nennen wir das Periodengitter von DL_{RQ}. Falls wir eine Basis von L finden, dann können wir das erweiterte DL-Problem leicht lösen.

Sei

$$\mathcal{Y} = \{\, \mathbf{y} \in \mathcal{P} \mid \frac{1}{16q}(y_1,y_2,y_3) = \mathbf{t} + \mathbf{e} \text{ mit } \mathbf{t} \in L^*,\ |e_1|,|e_2|,|e_3| \leq \frac{1}{32q} \text{ und } y_3 \leq \frac{q}{7} \,\}$$

Satz 5.7.4 *Seien $\Delta > 0$ eine Diskriminante, $\mathfrak{a}$, $\mathfrak{b}$ zwei reduzierte $\mathcal{O}_\Delta$-Ideale und L das Periodengitter von DL_{RQ} mit $\lambda_1(L) > 2^{11}$. Darüberhinaus sei q eine Zweierpotenz mit $q > 8\nu(L)$.*

Es gelten die folgenden Aussagen:

Algorithm 44 SampleDual-RQ

Input: q, $(\mathfrak{a}_1, d_1)$, …, $(\mathfrak{a}_{3\log_2 q}, d_{3\log_2 q})$.
Output: Vektor aus dem Gitter, das zum Periodengitter von f dual ist.

1. Initialzustand
$$|0\rangle|0\rangle|0\rangle|0\rangle$$
2. Erzeuge Superposition
$$\frac{1}{q\sqrt{q}} \sum_{x_1=0}^{q-1} \sum_{x_2=0}^{q-1} \sum_{x_3=0}^{q-1} |x_1\rangle|x_2\rangle|x_3\rangle|0\rangle$$
3. Berechne die Funktion
$$\frac{1}{q\sqrt{q}} \sum_{\mathbf{x}\in\mathcal{Q}} |\mathbf{x}\rangle|\operatorname{DL}_{RQ}(\mathbf{x})\rangle$$
4. Messe das letzte Register
$$\frac{1}{\sqrt{p}} \sum_{\mathbf{x}\in\mathcal{M}} |\mathbf{x}\rangle|\operatorname{DL}_{RQ}(\mathbf{x}')\rangle,$$
mit $\mathcal{M} = \{\, \mathbf{x} = (x_1, \ldots, x_r) \in \mathcal{Q} \mid f(\mathbf{x}) = f(\mathbf{x}') \,\}$ und $p = \operatorname{card} \mathcal{M}$.
5. Wende die Quantenfouriertransformation auf jedes der ersten drei Register an
$$\frac{1}{2^6 q\sqrt{qp}} \sum_{\mathbf{x}\in\mathcal{M}} \sum_{\mathbf{y}\in\mathcal{P}} \exp\left(2\pi i(\frac{x_1y_1}{16q} + \frac{x_2y_2}{16q} + \frac{x_1y_3}{16q})\right) |\mathbf{y}\rangle|f(\mathbf{x}')\rangle$$
6. Messe die ersten drei Register und gebe sie aus.

1. *Für einen Vektor* $\mathbf{y}$*, welcher am Ende des Algorithmus gemessen wird, ist die Wahrscheinlichkeit* $\Pr(\mathbf{y})$*, dass* $\mathbf{y} \in \mathcal{Y}$ *ist, mindestens* $1/(2^{16} \det L)$*,*

2. $\operatorname{card} \mathcal{Y} \geq \det L/256$

3. *Die Wahrscheinlichkeit, dass das abschließende Messen eine Approximation eines Vektors aus dem Gitter* L^* *ergibt, ist mindestens* 2^{-25}*.*

Beweis. Wir verwenden die Notation aus dem Satz und dem Algorithmus.

Wir beweisen zuerst die erste Aussage des Satzes. Sei

$$\mathcal{L} = \{\, \mathbf{u} \in L \mid \mathbf{x}' + \mathbf{u} + \mathbf{w}(\mathbf{u}) \in \mathcal{Q},\ \text{mit } \mathbf{w}(\mathbf{u}) = (0, 0, \epsilon_{(\mathbf{x},x_3')}) \text{ und } |\epsilon_{(\mathbf{x},x_3')}| \leq 1 \,\},$$

dann können wir, wegen Definition 5.7.2, die Menge $\mathcal{M}$ in Partitionen

$$\mathcal{M}_{\mathbf{u}} = \{\, \mathbf{m} \in \{0\}^2 \times \{0, \ldots, 6\} \mid \mathbf{x}' + \mathbf{u} + \mathbf{w}(\mathbf{u}) + \mathbf{m} \in \mathcal{Q},\ \text{mit } \mathbf{w}(\mathbf{u}) = (0, 0, \epsilon_{(\mathbf{x},x_3')}),\ |\epsilon_{(\mathbf{x},x_3')}| \leq 1 \text{ und } f(\mathbf{x}' + \mathbf{u} + \mathbf{w}(\mathbf{u})) = f(\mathbf{x}' + \mathbf{u} + \mathbf{w}(\mathbf{u}) + \mathbf{m}) \,\}.$$

aufteilen, mit $\mathbf{u} \in \mathcal{L}$.

Die Wahrscheinlichkeit, daß ein $\mathbf{y} \in \mathcal{Y}$ gemessen wird, ist

$$\begin{aligned}
\Pr(\mathbf{y}) &= \frac{1}{2^{12} q^2 qp} \left| \sum_{\mathbf{x} \in \mathcal{M}} \exp(2\pi i (\frac{x_1 y_1}{16q} + \frac{x_2 y_2}{16q} + \frac{x_3 y_3}{16q})) \right|^2 \\
&= \frac{1}{2^{12} q^2 qp} \left| \sum_{\mathbf{u} \in \mathcal{L}} \sum_{\mathbf{m} \in \mathcal{M}_{\mathbf{u}}} \exp(2\pi i (\frac{(x_1' + u_1) y_1 + (x_2' + u_2) y_2}{16q} + \frac{(x_3' + u_3 + m_3 + \epsilon_{(\mathbf{x},x_3')}) y_3}{16q})) \right|^2 \\
&= \frac{1}{2^{12} q^2 qp} \left| \sum_{\mathbf{u} \in \mathcal{L}} \sum_{\mathbf{m} \in \mathcal{M}_{\mathbf{u}}} \exp(2\pi i (\frac{u_1 y_1 + u_2 y_2}{16q} + \frac{(u_3 + m_3 + \epsilon_{(\mathbf{x},x_3')}) y_3}{16q})) \right|^2
\end{aligned}$$

mit $|\epsilon_{(\mathbf{x},x_3')}| \leq 1$. Wir schätzen nun die Mächtigkeit der Menge $\mathcal{L}$ von unten ab. Es gilt

$$\begin{aligned}
\operatorname{card} \mathcal{L} &\geq \{\, \mathbf{u} \in L \mid \mathbf{x}' + \mathbf{u} + \mathbf{w}(\mathbf{u}) \in \mathcal{Q} \,\} \geq \{\, \mathbf{x} \in L \mid \nu(L) \leq x_1, x_2 < q,\ 1 + \nu(L) \leq x_3 < q - 1 \,\} \\
&\geq (q - 3\nu(L))^2 (q - 3\nu(L) - 2) / \det L
\end{aligned}$$

Daraus erhalten wir für die Anzahl der Elemente der Menge $\mathcal{M}$ die folgende Schranke:

$$\operatorname{card} \mathcal{M} = \sum_{\mathbf{u} \in \mathcal{L}} \operatorname{card} \mathcal{M}_{\mathbf{u}} \geq \operatorname{card} \mathcal{L} \geq \frac{(q - 3\nu(L))^2 (q - 3\nu(L) - 2)}{\det L}$$

Sei nun $\mathbf{t} \in L^*$ der Gitterpunkt, der von $\mathbf{y}$ approximiert wird (siehe die Definition von $\mathcal{Y}$), dann gilt

$$\frac{u_1 y_1 + u_2 y_2 + (u_3 + m_3 + \epsilon_{(\mathbf{x},x_3')}) y_3}{16q} = \mathbf{u}(\mathbf{t}+\mathbf{e}) + \frac{(m_3 + \epsilon_{(\mathbf{x},x_3')}) y_3}{16q} \equiv \mathbf{ue} + \frac{(m_3 + \epsilon_{(\mathbf{x},x_3')}) y_3}{16q} \mod 1.$$

Außerdem gilt

$$\begin{aligned} 0 &\leq \max(\mathbf{ue} + \frac{(m_3 + \epsilon_{(\mathbf{x},x_3')})y_3}{16q}) - \min(\mathbf{ue} + \frac{(m_3 + \epsilon_{(\mathbf{x},x_3')})y_3}{16q}) \\ &\leq \frac{3}{32} + \frac{7}{16q}\frac{q}{7} + \frac{1}{16q}\frac{q}{7} \leq \frac{1}{4}. \end{aligned} \tag{5.7}$$

Daraus folgt nun

$$\begin{aligned} \Pr(\mathbf{y}) &= \frac{1}{2^{12}q^2qp}\left|\sum_{\mathbf{u}\in\mathcal{L}}\sum_{\mathbf{m}\in\mathcal{M}_\mathbf{u}} \exp(2\pi i(\frac{u_1y_1 + u_2y_2}{16q} + \frac{(u_3 + m_3 + \epsilon_{(\mathbf{x},x_3')})y_3}{16q}))\right|^2 \\ &= \frac{1}{2^{12}q^2qp}\left|\sum_{\mathbf{u}\in\mathcal{L}}\sum_{\mathbf{m}\in\mathcal{M}_\mathbf{u}} \exp(2\pi i(\mathbf{ue} + \frac{(m_3 + \epsilon_{(\mathbf{x},x_3')})y_3}{16q}))\right|^2 \\ &\geq \frac{p^2}{2^{12}q^2qp}|\sin 2\pi i\frac{1}{8}|^2 \geq \frac{(q - 3\nu(L))^2(q - 3\nu(L) - 2)}{2^{13}q^3\det L} \\ &> \frac{1}{2^{16}\det L} \end{aligned}$$

Die Ungleichung in der dritten Zeile folgt aus der Tatsache, dass in der Summe Vektoren in $\mathbb{C}$ aufsummiert werden, welche in einem Segment mit einem Winkel von höchstens $\pi/4$ liegen und deren Länge eins ist. Die letzte Ungleichung gilt wegen der Annahmen im Satz.

Als nächstes beweisen wir die zweite Aussage des Satzes. Es gilt nämlich

$$\begin{aligned} \operatorname{card}\mathcal{Y} &\geq \operatorname{card}\{\,\mathbf{t} = (t_1,t_2,t_3) \in L^* \mid \frac{1}{32q} \leq t_1,t_2 \leq 1 - \frac{1}{32q},\ \frac{1}{32q} \leq t_3 \leq \frac{1}{16\cdot 7} - \frac{1}{32q}\,\} \\ &\geq \frac{1}{\det L^*}\left(1 - \frac{1}{16q} - 2\nu(L^*)\right)^2\left(\frac{1}{112} - \frac{1}{16q} - 2\nu(L^*)\right) \\ &\geq \det L\left(1 - \frac{1}{16q} - \frac{6}{\lambda_1(L)}\right)^2\left(\frac{1}{112} - \frac{1}{16q} - \frac{6}{\lambda_1(L)}\right) \\ &\geq \frac{\det L}{2^8} \end{aligned}$$

Die dritte Aussage des Satzes folgt sofort aus den ersten zwei Aussagen des Satzes und dem Lemma 5.7.3. □

Satz 5.7.5 *Seien Δ die Diskriminante eines reell-quadratischen Zahlkörpers und $\mathfrak{a}$ ein $\mathcal{O}_\Delta$-Ideal. Bei der Eingabe von Δ, $\mathfrak{a}$ und $\mathfrak{b}$ berechnet* DL-RQ *den erweiterten diskreten Logarithmus von $\mathfrak{b}$ zur Basis $\mathfrak{a}$. Die Erfolgswahrscheinlichkeit des Algorithmus ist mindestens $1/2^{-164}$.*

Sei $n = \operatorname{size}(\Delta)$. Der Algorithmus benötigt für die internen Berechnungen höchstens $14n + 16\log_2^2 n + 93\log n + 252$ Qubits, falls die Quantenaddition verwendet wird, bzw. höchstens $16n + 16\log_2^2 n + 97\log n + 262$ Qubits, falls die klassische Addition verwendet wird.

Die Anzahl der elementaren Quantengatter ist $O(n^6)$, falls die klassische Addition verwendet wird, bzw. $O(n^7)$, falls die Quantenaddition verwendet wird.

Klassische Vor- und Nachberechnungen können in Polynomzeit durchgeführt werden □

Wir verzichten hier auf einen formalen Beweis, welcher analog zu den Beweisen von Satz 5.6.1 und Satz 5.6.1 verläuft.

Algorithm 45 DL-RQ

Input: Die Diskriminante Δ eines reell-quadratischen Zahlkörpers und zwei $\mathcal{O}_\Delta$-Ideale $\mathfrak{a}$ und $\mathfrak{b}$, so daß $\mathfrak{a}^n = \gamma\mathfrak{b}$, mit dem minimalen $n \in \mathbb{N}$ und einem $\gamma \in \mathcal{K}$ gilt.
Output: Der diskrete Logarithmus n und der Abstand $\mathrm{dist}(\mathfrak{a}^n, \mathfrak{b})$.

1. Setze $q = 2^x$, so daß $q/2 < 2^{53}\Delta\sqrt{\Delta}\ln^3\Delta \leq q$ gilt.
2. Berechne klassisch $f_i = \mathrm{Reg}(2^i)$ und $d_i \in \mathbb{Q}$ für $i = 1, \ldots, \log_2 q$, wobei $|\mathrm{dist}(\mathcal{O}_\Delta, f_i) - d_i - \left\lfloor 2^{i-2}/R \right\rfloor R + 2^{i-2}| \leq 1/(64\log_2 q)$ ist.
3. Berechne klassisch f_i' und d_i' für $i = 1, \ldots, \log_2 q$, so daß gilt: $f_i' = (a, b)$ ist das erste reduzierte Ideal mit $a > 0$ in der Reduktionssequenz von $\mathfrak{a}^i$ und $d_i' \in \mathbb{Q}$ mit $|\mathrm{dist}(\mathfrak{a}^i, f_i') - d_i'| \leq 1/(64\log_2 q)$.
4. Berechne klassisch g_i' und e_i' für $i = 1, \ldots, \log_2 q$, so daß gilt: $g_i' = (a, b)$ ist das erste reduzierte Ideal mit $a > 0$ in der Reduktionssequenz von $\mathfrak{b}^i$ und $e_i' \in \mathbb{Q}$ mit $|\mathrm{dist}(\mathfrak{b}^i, g_i') - d_i'| \leq 1/(64\log_2 q)$.
5. Führe sechsmal SampleDual-RQ aus. Seien $\mathbf{y}_i \in \mathbb{Z}^2$, $i = 1, \ldots, 6$ die zurückgegebenen Paare.
6. Mit Algorithmus aus Satz 2.4.24 berechne eine Basis des Periodengitters der Funktion DL_{RQ}.
7. Berechne n und S aus der oben bestimmten Basis.
8. Mit klassischen Algorithmen (siehe [BB94], [Thi95] oder [Mau00]) bestimme den Abstand S mit der geforderten Approximationsgüte.
9. Gebe n und S aus.

Kapitel 6

Algorithmen für Zahlkörper beliebigen Grades

In diesem Kapitel wird ein Quantenalgorithmus zur Berechnung der Einheitengruppe eines Zahlkörpers eines beliebigen Grades entwickelt und untersucht. Es wird gezeigt, daß die Laufzeit des Algorithmus polynomiell von der Diskriminante des Zahlkörpers und exponentiell vom Körpergrad abhängt.

Das Kapitel ist in zwei Abschnitte unterteilt. Im ersten Abschnitt wird ein Quantenalgorithmus zur Berechnung des Periodengitters einer schwach-periodischen Funktion präsentiert und seine Laufzeit und die Erfolgswahrscheinlichkeit untersucht. Im zweiten Teil wird eine schwach-periodische Funktion beschrieben, aus deren Periodengitter sich die Einheitengruppe eines Zahlkörpers bestimmen läßt.

6.1 Hidden-Subgroup-Problem in der Gruppe $\mathbb{R}^r$

In diesem Abschnitt präsentieren wir einen Algorithmus zur Berechnung einer Approximation einer Basis des Periodengitters einer schwach-periodischen Funktion. Wir bestimmen die Laufzeit und den Speicherbedarf des Algorithmus und geben eine untere Grenze für die Erfolgswahrscheinlichkeit an.

Wir fangen mit der Definition einer schwach-periodischen Funktion an.

Satz 6.1.1 (Schwach-periodische Funktion) *Seien $r \in \mathbb{N}$, $\mathcal{S}$ eine Menge und $f : \mathbb{Z}^r \longrightarrow \mathcal{S}$ eine Funktion. Dann heißt f schwach-periodisch mit Periodengitter $L \subset \mathbb{R}^r$ in einem Punkt $\mathbf{x} \in \mathbb{Z}^r$, wenn für alle $\mathbf{y} \in L$ genau ein $\mathbf{e} \in \mathbb{R}^r$ mit $\|\mathbf{e}\|_\infty \leq 2$ existiert, so daß $f(\mathbf{x}) = f(\mathbf{x} + \mathbf{y} + \mathbf{e})$.*

Der Quantenalgorithmus zur Berechnung einer Approximation des Periodengitters einer schwach-periodischen Funktion f ist auf der folgenden Seite abgebildet.

Als nächstes analysieren wir die angegebene Algorithmen. Zuerst berechnen wir die Erfolgswahrscheinlichkeit des Algorithmus HSP-MD. Dafür berechnen wir die Wahrscheinlichkeit

Algorithm 46 SampleDual-MD

Input: $q, r \in \mathbb{N}$ und eine schwach-periodische Funktion f.
Output: Vektor aus dem Gitter, das zum Periodengitter von f dual ist.

1. Initialzustand
$$|0\rangle \dots |0\rangle.$$
2. Erzeuge Superposition
$$\frac{1}{q^{r/2}} \sum_{x_1=0}^{q-1} \cdots \sum_{x_r=0}^{q-1} |x_1\rangle \dots |x_r\rangle |0\rangle.$$
3. Berechne die Funktion
$$\frac{1}{q^{r/2}} \sum_{x_1=0}^{q-1} \cdots \sum_{x_r=0}^{q-1} |x_1\rangle \dots |x_r\rangle |f(\mathbf{x})\rangle.$$
4. Messe das letzte Register
$$\frac{1}{\sqrt{p}} \sum_{\mathbf{x} \in \mathcal{M}}^{q-1} |\mathbf{x}\rangle |f(\mathbf{x}')\rangle$$
mit $\mathcal{M} = \{ \mathbf{x} = (x_1, \dots, x_r) \in \mathbb{Z}^r \mid f(\mathbf{x}) = f(\mathbf{x}') \text{ und } 0 \leq x_i < q \text{ für } 1 \leq i \leq r \}$ und $p = \operatorname{card} \mathcal{M}$.
5. Wende Quantenfouriertransformation auf jedes der ersten r Register an
$$\frac{1}{\sqrt{(2rq)^r p}} \sum_{y_1=0}^{2rq-1} \cdots \sum_{y_r=0}^{2rq-1} \sum \mathbf{x} \in \mathcal{M} \exp(2\pi i \frac{\mathbf{x} \cdot \mathbf{y}}{2rq}) |y_1\rangle \dots |y_r\rangle |f(\mathbf{x}')\rangle.$$
6. Messe die ersten r Register und gebe sie aus.

Algorithm 47 HSP-MD

Input: Eine schwach-periodische Funktion f
Output: Approximation einer Basis des Periodengitters

```
/* Mit einem Quantencomputer */
for i = 0, ..., 2r + 1 do
  a'_i ← SampleDual-MD(f)

/* Mit einem klassischen Computer */
C' ← FindDualBasis(a'_1, ..., a'_{2r+1})
Gebe aus: C'
```

$\Pr(\mathbf{y})$ dafür, daß Algorithmus SampleDual-MD eine "gute" Approximation $\mathbf{y}$ eines Vektors $\mathbf{z} \in L^*$ zurückgibt. Dabei bedeutet eine "gute" Approximation, daß $\mathbf{y}$ ein Element der Menge

$$\mathcal{Y} = \{\, \mathbf{y} \in \mathbb{N}^r \mid \frac{1}{2rq}\mathbf{y} = \mathbf{t} + \mathbf{e} \text{ mit } \mathbf{t} \in L^*,\ \|\mathbf{e}\|_\infty \leq \frac{1}{4rq},\ \text{und } \|\mathbf{y}\|_\infty \leq \frac{q}{32} \,\} \tag{6.1}$$

ist.

Lemma 6.1.2 *Wenn der Algorithmus* SampleDual-MD*, angewandt auf eine schwach-periodische Funktion f mit Periodengitter L und $\lambda_1(L) > 256r^2$, terminiert, dann gelten die folgenden drei Aussagen:*

1. *Die Wahrscheinlichkeit $\Pr(\mathbf{y})$, daß ein Vektor $\mathbf{y} \in \mathcal{Y}$ gemessen wird, ist mindestens $1/(2^{2r+2}r^r \det L)$,*
2. *$\operatorname{card} \mathcal{Y} \geq \det L/(256r)^r$,*
3. *Die Wahrscheinlichkeit, daß das abschließende Messen eine Approximation eines Vektors aus dem dualen Gitter L^* ergibt, ist mindestens $1/(2^{10r+2}r^{2r})$.*

Beweis. Wir beweisen zuerst die erste Aussage des Lemmas. Die Wahrscheinlichkeit ein $\mathbf{y} \in \mathcal{Y}$ zu messen ist

$$\begin{aligned}\Pr(\mathbf{y}) =& \frac{1}{(2rq)^r p}\left|\sum_{\mathbf{x}\in\mathcal{M}} \exp(2\pi i \frac{\mathbf{xy}}{2rq})\right|^2 = \frac{1}{(2rq)^r p}\left|\sum_{\mathbf{z}\in\mathcal{L}} \exp(2\pi i \frac{(\mathbf{x}' + \mathbf{z} + \mathbf{w}(\mathbf{z}))\cdot\mathbf{y}}{2rq})\right|^2 = \\ & \frac{1}{(2rq)^r p}\left|\sum_{\mathbf{z}\in\mathcal{L}} \exp(2\pi i \frac{(\mathbf{z} + \mathbf{w}(\mathbf{z}))\cdot\mathbf{y}}{2rq})\right|^2\end{aligned} \tag{6.2}$$

mit

$$\mathcal{L} = \{\, \mathbf{z} \in L \mid \text{es existiert ein } \mathbf{w}(\mathbf{z}) \in \mathbb{R}^r \text{ mit } \|\mathbf{w}(\mathbf{z})\|_\infty \leq 2 \text{ und } \mathbf{x}' + \mathbf{z} + \mathbf{w}(\mathbf{z}) \in \mathcal{M} \,\},$$

und es gilt

$$\frac{(\omega-2)^r \nu(L)^r}{\det L} \leq \operatorname{card}\mathcal{L} \leq \frac{(\omega+2)^r \nu(L)^r}{\det L}. \tag{6.3}$$

Wegen

$$\frac{1}{2rq}(\mathbf{z} + \mathbf{w}(\mathbf{z}))\cdot\mathbf{y} = (\mathbf{z} + \mathbf{w}(\mathbf{z}))\cdot(\mathbf{t} + \mathbf{e}) \equiv \mathbf{z}\cdot\mathbf{e} + h(\mathbf{z},\mathbf{y}) \pmod 1$$

mit $|h(\mathbf{z},\mathbf{y})| = |\mathbf{w}(\mathbf{z})\cdot(\mathbf{t}+\mathbf{e})| \leq \|2\frac{1}{2rq}\mathbf{z}\|_1 \leq \frac{1}{32}$ läßt sich (6.2) zu

$$\Pr(\mathbf{y}) = \frac{1}{(2rq)^r p}\left|\sum_{\mathbf{z}\in\mathcal{L}} \exp(2\pi i(\mathbf{z}\cdot\mathbf{e} + h(\mathbf{z},\mathbf{y})))\right|^2$$

umformen. Die Summe in der letzten Gleichung ist die Summe von $\operatorname{card}\mathcal{L}$ Vektoren in $\mathbb{C}$ der Länge Eins und einem Winkel dazwischen, der höchstens $3\pi/16$ beträgt. Daraus folgt, daß

$$\begin{aligned}\Pr(\mathbf{y}) &\geq \frac{p^2}{(2rq)^r p} * |\sin\frac{3\pi}{16}|^2 \geq \frac{p}{2^{r+2}(rq)^r} \geq \frac{(\omega-2)^r\nu(L)^r}{2^{r+2}(rq)^r \det L} \\ &\geq \frac{(\omega-2)^r}{\omega^r}\frac{1}{2^{r+2}r^r \det L} \geq \frac{1}{2^{2r+2}r^r \det L}\end{aligned}$$

Als nächstes beweisen wir die zweite Aussage des Lemmas. Es gilt

$$\begin{aligned}\operatorname{card}\mathcal{Y} &= \operatorname{card}\{\,\mathbf{y}\in\mathbb{N}^r \mid \frac{1}{2rq}\mathbf{y} = \mathbf{t}+\mathbf{e} \text{ mit } \mathbf{t}\in L^*,\ \|\mathbf{e}\|_\infty \le \frac{1}{4rq},\ \text{und } \|\mathbf{y}\|_\infty \le \frac{q}{32}\,\}\\ &\ge \operatorname{card}\{\,\mathbf{t}=(t_1,\ldots,t_r)\in L^* \mid \frac{1}{4rq}\le t_i \le \frac{1}{64r}-\frac{1}{4rq}\,\}\\ &\ge \frac{1}{\det L^*}\left(\frac{1}{64r}-\frac{1}{2rq}-2\nu(L^*)\right)^r \ge \det L\left(\frac{1}{64r}-\frac{1}{2rq}-\frac{2r}{\lambda_1(L)}\right)^r\\ &\ge \frac{\det L}{(256r)^r},\end{aligned}$$

da wir angenommen haben, daß $\lambda_1(L)\ge 256r^2$ und $q>128$.

Die letzte Aussage des Lemmas folgt sofort aus den ersten zwei. □

Wir führen nun die Ergebnisse aus dem letzten Lemma und dem Sätzen 2.4.23 und 2.4.24 zum folgenden Satz zusammen:

Satz 6.1.3 *Seien $r,q\in\mathbb{N}$, $\mathcal{S}$ eine Menge und $L\in\mathbb{R}^r$ ein vollständiges Gitter mit $\lambda_1(L)\ge 2^{2r+7}r^{(r+3)/2}$. Sei $f:\mathbb{Z}^r\longrightarrow\mathcal{S}$ eine schwach-periodische Funktion mit Periodengitter L, für welche gilt*

$$\begin{aligned}\operatorname{card}\{\,\mathbf{x}\in\mathbb{Z}^r \mid 0\le x_i<q,\ 0\le i\le r \text{ und } f \text{ ist}\\ \text{schwach-periodisch in } \mathbf{x} \text{ mit Periodengitter } L\,\} \ge \xi q^r\end{aligned} \tag{6.4}$$

mit $\xi>0$. Sei außerdem

$$q=\max\{128,\omega\nu(L)\},\quad \text{mit } \omega\ge 2^{2r^2}(\det L)^3. \tag{6.5}$$

Dann berechnet der Algorithmus HSP-MD *Vektoren $\mathbf{c}'_1,\ldots,\mathbf{c}'_r\in\mathbb{Q}^r$, die eine Basis $\mathbf{c}_1,\ldots,\mathbf{c}_r$ von L approximieren. Dabei gilt*

$$\|\mathbf{c}'_j-\mathbf{c}\|_\infty\le 1,\quad 0\le j\le r. \tag{6.6}$$

Sei $T(f,q)$ die Zeit, die benötigt wird, um die Funktion f mit Parametern kleiner als q auszuwerten. Dann ist die Laufzeit des Algorithmus $O(rT(f,q)+r^7\log^3 q)$. Die Erfolgswahrscheinlichkeit ist mindestens $\xi^{2r}/(2^{20r^2+8r+2}r^{4r^2})$.

Beweis. Wir verwenden die gleichen Bezeichnungen wie im Algorithmus und im Satz.

Als erstens beweisen wir die Aussage über die Erfolgswahrscheinlichkeit des Algorithmus. Diese Wahrscheinlichkeit hängt davon ab, daß SampleDual-MD gute Approximationen von Vektoren aus dem Gitter L^* zurückgibt und daß diese Vektoren das komplette Gitter L^* erzeugen.

Aus Lemma 6.1.2, (6.4) und $\lambda_1(L)\ge 2^{2r+7}r^{(r+3)/2}\ge 256r^2$ folgt, daß die Wahrscheinlichkeit, eine Approximation $\mathbf{a}'$ eines Vektor $\mathbf{a}\in L^*$ zu messen, mindestens $\xi/(2^{10r+2}r^{2r})$ ist.

Aus Theorem 2.4.11 und $\lambda_1(L)\ge 2^{2r+7}r^{(r+3)/2}$ folgt, daß $\nu(L^*)\le 2^{-2r-7}r^{(r+1)/2}$ gilt. Somit ist die Bedingung im Satz 2.4.23 erfüllt. Demzufolge ist die Wahrscheinlichkeit, daß $2r$ Vektoren $\mathbf{a}_1,\ldots,\mathbf{a}_{2r}$ das gesamte Gitter L^* und die ersten r Vektoren das r-dimensionale Untergitter erzeugen, mindestens $1/2^{4r+2}$. Deshalb ist die Erfolgswahrscheinlichkeit von HSP-MD mindestens

$$\left(\frac{\xi}{2^{10r+2}r^{2r}}\right)^{2r}\frac{1}{2^{4r+2}}\ge\frac{\xi^{2r}}{2^{20r^2+8r+2}r^{4r^2}}.$$

Als nächstes zeigen wir, daß die Wahl von q ausreichend ist, um die Präzision wie in (6.6) zu erreichen. Dazu verwenden wir Satz 2.4.24. Wegen (6.1) gilt

$$\|\mathbf{a}' - 2rq\mathbf{a}\|_2 \leq \sqrt{r}\|\mathbf{a}' - 2rq\mathbf{a}\|_\infty \leq \sqrt{r}2rq\|\mathbf{e}\|_\infty \leq \frac{\sqrt{r}}{2}.$$

für alle gemessenen Vektoren $\mathbf{a} \in \mathcal{Y}$. Deshalb ist die Annahme (2.4) mit $h = \log_2(2rq)$ in Satz 2.4.24 erfüllt. Wir setzen $M = L^*$, $\epsilon = 1$, $\alpha = 1/(64r)$, $\lambda_{max} = 1$ und $\mu = \lambda_1(L^*)$, um auch die Annahmen (2.3) zu erfüllen. Aus Theorem 2.4.11 folgt, daß $\mu \leq 1/\nu(L)$ ist, so daß zusammen mit (6.5) die Ungleichung (2.4) wahr wird. Deshalb können wir den Algorithmus aus Satz 2.4.24 auf die Vektoren $\mathbf{a}'_1, \ldots, \mathbf{a}'_{2r}$ anwenden und bekommen eine Approximation einer Basis von L, die (6.6) erfüllt.

Zum Schluß beweisen wir die Aussage über die Laufzeit von HSP-MD. Der Algorithmus wertet $2r$ mal die Funktion f aus und führt $2r$ mal die Quantenfouriertransformation auf r Quantenregistern der Größe $\log_2(2rq)$ aus. Anschließend wird klassisch die Basis des Gitters L berechnet. Sei M_r ein Untergitter von L^*, welches von den ersten r Vektoren, welche die Funktion SampleDual-MD berechnet, erzeugt wird. Dann ist die Gesamtlaufzeit des Algorithmus

$$\begin{aligned}&O((2r)(T(f,q) + r\log_2^2(2rq))) + O(r^6\log^3(2^h \max\{\alpha, r^{\frac{3}{2}}\lambda_r(M_r)\})) = \\ &O(rT(f,q) + r\log_2^2(2rq) + r^6\log^3(2rqr^{\frac{3}{2}})) = O(rT(f,q) + r^7\log^3 q).\end{aligned}$$

□

6.2 Berechnung von Einheiten

In diesem Abschnitt definieren wir eine schwach-periodische Funktion f mit dem Periodengitter $(N \operatorname{Log} U)$ (die Zahl $N \in \mathbb{N}$ wird später definiert, U ist die Einheitengruppe). Aus diesem Gitter können wir die Einheitengruppe eines Zahlkörpers leicht bestimmen.

Die Funktion f bildet einen Vektor $\mathbf{v} \in \mathbb{Z}^r$ auf das zu $\mathbf{v}$ nächste Minimum μ und den Abstand zum Vektor $\operatorname{Log}\mu$ ab. Der Abstand ist nötig, um die Injektivität der Funktion innerhalb einer Periode zu erreichen, welche für den Quantenalgorithmus aus dem letzten Abschnitt erforderlich ist. Wir verwenden dabei den Algorithmus aus CollectMinima (siehe Satz 2.5.44), welcher uns alle Minima in der Nähe des Punktes $\mathbf{v}$ berechnet. Aus diesen Minima wählen wir dann das nächste aus und berechnen den Abstand.

Dabei treten jedoch zwei Probleme auf. Das erste Problem ist, das nächste Minimum zu einem Punkt $\mathbf{v}$ zu bestimmen. Dafür verwenden wir die Funktion $\operatorname{Log} : \mathcal{K} \longrightarrow \mathbb{R}^r$. Diese Funktion kann jedoch nur approximativ berechnet werden. Das führt dazu, daß manchmal nicht entschieden werden kann, ob ein Minimum wirklich das nächste zu $\mathbf{v}$ ist.

Das zweite Problem ist, daß der Algorithmus CollectMinima die binäre multiplikative Darstellung (BMD) der Minima zurückgibt. Die BMDs sind jedoch nicht eindeutig, sondern hängen von $\mathbf{v}$ ab. Deshalb kann es vorkommen, daß wir bei zwei benachbarten Punkten $\mathbf{v}, \mathbf{v}' \in \mathbb{Z}^r$, $\|\mathbf{v} - \mathbf{v}'\|_\infty = 1$, mit dem gleichen nächsten Minimum μ, bei der Berechnung der Abstände $\mathbf{d} = \mathbf{v} - \text{ApproxLog}(\mu)$ bzw. $\mathbf{d}' = \mathbf{v}' - \text{ApproxLog}(\mu)$ entweder gleiche Vektoren $(\mathbf{d} = \mathbf{d}')$ oder Vektoren mit $\|\mathbf{d} - \mathbf{d}'\|_\infty > 1$ erhalten. Das erste führt dazu, daß die Injektivität

innerhalb einer Periode verloren geht, während das zweite zu Lücken im Periodengitter der Funktion führt.

6.2.1 Eindeutige binäre multiplikative Darstellung

Wir entwickeln nun ein Algorithmus, der die beiden oben genannten Probleme löst und zu jedem Minimum eine eindeutige BMD berechnet. Wir nennen diesen Algorithmus DistMBD.

Als erstes beschreiben wir, wie wir zu einem Minimum μ, welches in der binären multiplikativen Darstellung $\mathbf{b}(\mu)$ gegeben ist, einen eindeutigen Punkt $\mathbf{w} \in \frac{1}{N}\mathbb{Z}^r$ mit $\|\mathbf{w} - \textsc{ApproxLog}(\mathbf{b}(\mu))\|_2 < \delta$ zuordnen, welcher unabhängig von der gegebenen BMD ist. Dazu berechnen wir zuerst $\mathbf{t} = \textsc{ApproxLog}(\mathbf{b}(\mu))$. Aus dem Satz 2.5.42 folgt, daß

$$t_i - \delta < \ln|\mu|_i \leq t_i, \quad \text{für } 1 \leq i \leq r. \tag{6.7}$$

Sei

$$\mathcal{N}(\mathbf{t}) = \{\, \mathbf{w} \in \frac{1}{N}\mathbb{Z}^r \mid t_i - \delta - \frac{1}{N} < w_i \leq t_i + \frac{1}{N},\ 1 \leq i \leq r \,\}.$$

Die Menge $\mathcal{N}(\mathbf{t})$ enthält unter anderem alle Punkte aus $\frac{1}{N}\mathbb{Z}^r$ mit dem euklidischem Abstand zu $\operatorname{Log}\mu$ von höchstens δ, d.h.

$$\{\, \textsc{ApproxLog}(\mathbf{b}) \mid \mathbf{b} \text{ ist eine BMD von } \mu \,\} \subseteq \mathcal{N}(\mathbf{t}).$$

Wir wenden CollectMinima auf jeden Punkt $\mathbf{u} \in \mathcal{N}(\mathbf{t})$ an und berechnen zu jedem $\mathbf{u}$ eine BMD von μ. Wir bezeichnen diese Darstellung mit $\mathbf{b}(\mathbf{u})$. Schließlich wählen wir den Punkt $\mathbf{w} \in \mathcal{N}(\mathbf{t})$ aus, so daß für alle $\mathbf{u} \in \mathcal{N}(\mathbf{t})$ mit $\mathbf{u} \neq \mathbf{w}$ entweder

$$\|\mathbf{w} - \textsc{ApproxLog}(\mathbf{b}(\mathbf{w}))\|_2 < \|\mathbf{w} - \textsc{ApproxLog}(\mathbf{b}(\mathbf{u}))\|_2 \tag{6.8}$$

oder

$$\|\mathbf{w} - \textsc{ApproxLog}(\mathbf{b}(\mathbf{w}))\|_2 = \|\mathbf{w} - \textsc{ApproxLog}(\mathbf{b}(\mathbf{u}))\|_2 \text{ und } \mathbf{w} <_{\text{lex}} \mathbf{u} \tag{6.9}$$

gilt, wobei $<_{\text{lex}}$ für eine lexikographische Ordnung steht.

Lemma 6.2.1 *Sei $\delta < 1/N$ so gewählt, daß $(-\log\delta)$ höchstens polynomiell in $\log|\Delta|$ ist. Sei außerdem μ ein Minimum, welches in der BMD $\mathbf{b}$ gegeben ist. Bei Eingabe von $\mathbf{b}$ berechnet* DistMBD *eine eindeutige, von $\mathbf{b}$ unabhängige BMD von μ. Die Laufzeit des Algorithmus ist polynomiell in $\log|\Delta|$ und exponentiell im Körpergrad.*

Beweis. Wir beweisen hier die Aussage über die Laufzeit des Algorithmus. Wie oben beschrieben, führt DistMBD einmal den Algorithmus ApproxLog und $|\mathcal{N}(\mathbf{t})|$ mal den Algorithmus CollectMinima aus. Laut Satz 2.5.42 ist die Laufzeit von ApproxLog polynomiell in $\log|\Delta|$. Durch die Wahl von $\delta < 1/N$ enthält Menge $\mathcal{N}(\mathbf{t})$ polymomiell viele Elemente. Satz 2.5.44 besagt, daß die Laufzeit von CollectMinima polynomiell in $\log|\Delta|$ und exponentiell im Körpergrad ist. Daraus folgt, daß auch die Laufzeit vom gesamten Algorithmus polynomiell in $\log|\Delta|$ und exponentiell im Köpergrad ist. □

6.2.2 Schwach-periodische Funktion

Nun beschreiben wir, wie mit Algorithmus DistMBD die Funktion f konstruiert werden kann.

Satz 6.2.2 *Sei $\mathbf{v} \in \frac{1}{N}\mathbb{Z}^r$ ein Vektor. Dann ist ein Minimum $\mu = \mu(\mathbf{v}) \in \mathcal{M}_{\mathcal{O}}$ genau dann das zu* $\mathbf{v}$ nächste Minimum*, wenn für alle anderen Minima $\mu' \in \mathcal{M}_{\mathcal{O}}$ gilt*

$$\|\mathbf{v} - \textsc{ApproxLog}(\textsc{DistMBD}(\mu))\|_2 < \|\mathbf{v} - \textsc{ApproxLog}(\textsc{DistMBD}(\mu'))\|_2$$

oder

$$\|\mathbf{v} - \textsc{ApproxLog}(\textsc{DistMBD}(\mu))\|_2 = \|\mathbf{v} - \textsc{ApproxLog}(\textsc{DistMBD}(\mu'))\|_2 \text{ und}$$
$$\textsc{ApproxLog}(\textsc{DistMBD}(\mu)) <_{\text{lex}} \textsc{ApproxLog}(\textsc{DistMBD}(\mu')).$$

Wir definieren f folgendermaßen:

$$f_N : \mathbb{Z}^r \longrightarrow \mathcal{R} \times \mathbb{Z}^r : \mathbf{v} \longmapsto ((1/\mu), \lceil N(\mathbf{v}/N - \mathbf{L}(\mu))\rceil),$$

wobei μ das zu $\mathbf{v}/N$ nächste Minimum ist. Und zeigen in den zwei nachfolgenden Lemmata, daß es eine große Anzahl von Vektoren in $\mathbb{Z}^r$ gibt, in welchen die Funktion schwach-periodisch ist, vorausgesetzt die Zahl N ist groß genug gewählt. Für die Berechnung der Funktion verwenden wir den Algorithmus ApproxLog, deshalb müssen wir den Präzisionsparameter δ (siehe Satz 2.5.42) wählen. Sei nun $\delta \in \mathbb{Q}$ mit

$$\delta < \frac{1}{2N}.$$

Lemma 6.2.3 *Sei $\mathbf{v} \in \mathbb{Z}^r$. Wenn ein $\mu \in \mathcal{M}_{\mathcal{O}}$ existiert, so daß für jedes $\mu' \in \mathcal{M}_{\mathcal{O}}$*

$$\|\mathbf{v}/N - \textsc{ApproxLog}(\textsc{DistMBD}(\mu))\|_2 + 6\sqrt{r}/N < \|\mathbf{v}/N - \textsc{ApproxLog}(\textsc{DistMBD}(\mu'))\|_2 \tag{6.10}$$

gilt, dann ist f_N schwach-periodisch in $\mathbf{v}$ mit Periodengitter $\mathbb{N}\operatorname{Log} U$.

Beweis. Seien $\mathbf{v} \in \mathbb{Z}^r$ und $\mu \in \mathcal{M}_{\mathcal{O}}$, so daß (6.10) gilt. Seien ϵ eine Einheit, $\lambda = N \operatorname{Log} \epsilon$ und $\mathbf{w} \in \mathbb{R}^r$ mit $\|\mathbf{w}\|_\infty < 2$ und $\mathbf{v} + \lambda + \mathbf{w} \in \mathbb{Z}^r$. Wegen (6.10) gilt für alle Minima $\mu' \in \mathcal{M}_{\mathcal{O}}$, $\mu \neq \mu'$, die folgende Ungleichung:

$$\begin{aligned}
&\|\mathbf{v}/N + \lambda + \mathbf{w}/N - \textsc{ApproxLog}(\textsc{DistMBD}(\epsilon\mu))\|_2 \leq \\
&\|\mathbf{v}/N + \operatorname{Log}\epsilon - \operatorname{Log}(\epsilon\mu)\|_2 + \|\mathbf{w}/N - \delta\|_2 \leq \\
&\|\mathbf{v}/N + \operatorname{Log}\mu\|_2 + \frac{3\sqrt{r}}{N} \quad < \quad \|\mathbf{v}/N + \operatorname{Log}\mu'\|_2 - \frac{3\sqrt{r}}{N} \leq \\
&\|\mathbf{v}/N + \operatorname{Log}\epsilon - \operatorname{Log}(\epsilon\mu')\|_2 - \|\mathbf{w}/N - \delta'\|_2 \leq \\
&\|\mathbf{v}/N + \lambda + \mathbf{w}/N - \textsc{ApproxLog}(\textsc{DistMBD}(\epsilon\mu'))\|_2
\end{aligned}$$

mit δ, δ' aus dem Satz 2.5.42. D.h. $\epsilon\mu$ ist das zu $(\mathbf{v} + \lambda + \mathbf{w})/N$ nächste Minimum.

Aus dem Satz 2.5.42 folgt außerdem

$$\|\mathrm{Log}\,\epsilon - (\textsc{ApproxLog}(\textsc{DistMBD}(\epsilon\mu)) - \textsc{ApproxLog}(\textsc{DistMBD}(\mu)))\|_\infty \leq 2\delta < 1/N.$$

Daraus folgern wir, daß

$$\|\lceil \mathbf{v} + \lambda - N\textsc{ApproxLog}(\textsc{DistMBD}(\epsilon\mu))\rceil - \lceil \mathbf{v} - N\textsc{ApproxLog}(\textsc{DistMBD}(\mu))\rceil\|_2 \leq 2$$

ist und daß ein eindeutiges $\mathbf{w} \in \mathbb{R}^r$ mit $\mathbf{v} + \lambda + \mathbf{w} \in \mathbb{Z}^r$ und

$$\lceil (\mathbf{v} + \lambda + \mathbf{w}) - N\textsc{ApproxLog}(\textsc{DistMBD}(\epsilon\mu))\rceil = \lceil \mathbf{v} - N\textsc{ApproxLog}(\textsc{DistMBD}(\mu))\rceil$$

existiert. □

Sei

$$N(r, \Delta) = 3^2 2^{5r+2} r(\log|\Delta|)^{4r}|\Delta|^{n^2+1},$$

wobei n der Körpergrad ist.

Lemma 6.2.4 *Seien $\mathbf{v} \in \mathbb{R}^r$ ein Vektor und $N > N(r, \Delta)$ eine natürliche Zahl. Setze*

$$\mathcal{Q} = \mathcal{Q}(v) = \{\, \mathbf{w} \in \mathbb{Z}^r \mid 0 \leq w_i - v_i < \frac{N \log|\Delta|}{8},\ 1 \leq i \leq r \,\}$$

und

$$\mathcal{Q}^+ = \mathcal{Q}^+(\mathbf{v}) = \{\, \mathbf{w} \in \mathbb{Q}(\mathbf{v}) \mid \exists \mu \in \mathcal{M}_\mathcal{O}.\forall \mu' \in \mathcal{M}_\mathcal{O}.\|\mathbf{w}/N - \mathbf{L}(\mu)\|_2 + 6\sqrt{r}/N < \|\mathbf{w}/N - \mathbf{L}(\mu')\|_2 \,\},$$

dann gilt

$$\mathrm{card}\,\mathcal{Q}^+ / \mathrm{card}\,\mathcal{Q} > 2/3$$

.

Beweis. Sei $\mathbf{w} \in \mathcal{Q}\backslash\mathcal{Q}^+$. Wähle zwei Minima μ und μ', für welche $\textsc{ApproxLog}(\mu)$ und $\textsc{ApproxLog}(\mu')$ den minimalen Abstand zu $\mathbf{w}/N$ in der euklidischen Norm haben und für welche die Bedingung (6.10) verletzt ist. Dann liegt $\mathbf{w}/N$ zwischen zwei Hyperebenen, welche senkrecht zu $\textsc{ApproxLog}(\mu) - \textsc{ApproxLog}(\mu')$ liegen und deren Abstand kleiner als $3r \log\Delta\Delta^{n^2+1}/N$ ist. Das folgt aus der Tatsache, daß der euklidische Abstand zwischen den Bildern von zwei Minima unter der Dirichlet-Abbildung mindestens Δ^{-n^2-1} ist.

Die Menge aller Punkte in $\mathcal{Q}$, die zwischen diesen zwei Hyperebenen liegen und deren Abstand zu μ kleiner als $1/4 \log\Delta$ ist, ist enthalten in einem Körper, dessen Volumen kleiner als $2^{-2r}3r(\log\Delta)^r\Delta^{n^2+1}/N$ ist. Wir nennen diesen Körper $\mathcal{T}(\mu, \mu')$.

Die Minima μ und μ' sind in einem Hyperkubus mit Kantenlänge höchstens $\log\Delta/4$ enthalten. Ein Resultat von Buchmann besagt, daß die Anzahl der Minima innerhalb eines solchen Hyperkubus höchstens $2^{2r+2}(\log\Delta)^{2r}$ ist (siehe [Buc87]).

Die Addition der Volumen von $\mathcal{T}(\mu, \mu')$, wobei (μ, μ') alle möglichen Paare von Minima durchlaufen, impliziert die Aussage des Lemmas. □

Algorithm 48 UnitGroup

Input: Eine Ordnung $\mathcal{O}$ eines Zahlkörpers $\mathcal{K}$ vom Grad n über $\mathbb{Q}$.
Output: Fundamentaleinheiten von $\mathcal{K}$

1. Setze $N = \left\lceil \max\{N(r,\Delta), 2^{2r+18} r^{(r+5)/2}\} \right\rceil$, $\delta = 1/(3N)$ und $q = 3rN$.
2. Berechne die Vektoren $\{\mathbf{b}_1, \ldots, \mathbf{b}_r\}$ mit dem Algorithmus HSP-MD.
3. Mit Algorithmus CollectMinima stelle fest, ob alle b_i nahe an Minima in L liegen
4. Falls alle b_i nahe an Minima liegen, berechne mit CollectMinima die dazu gehörenden Einheiten.
5. Gebe die berechneten Einheiten aus.

Satz 6.2.5 *Sei U die Einheitengruppe der Ordnung $\mathcal{O}$ und $L = \{ \operatorname{Log} \alpha \mid \alpha \in U \}$. Für alle $N > N(r,\Delta)$ und alle Teilmengen $\mathcal{Q} \in \mathbb{R}^r$, wobei $\mathcal{Q}$ ein r-dimensionaler Hyperkubus mit Kantenlänge größer als $3rN$ ist, ist die Funktion f_N schwach-periodisch mit dem Periodengitter NL in mehr als der Hälfte der Punkte aus $\mathbb{Z}^r \cup \mathcal{Q}$.*

Die Laufzeit von f_N ist polynomiell in $\log|\Delta|$ und exponentiell im Körpergrad.

Beweis. Die Aussage folgt aus dem Lemma 6.2.3 und 6.2.4.

Nun kommen wir zum Beweis des Hauptsatzes in diesem Kapitel.

Satz 6.2.6 *Sei $\mathcal{O}$ eine Ordnung eines Zahlkörpers $\mathcal{K}$ vom Grad n über $\mathbb{Q}$ mit einer Diskriminante, deren Absolutbetrag Δ ist. Seien r der Einheitenrang und $\epsilon > 0$. Algorithmus* UnitGroup *berechnet bei Eingabe von $\mathcal{O}$ die Einheiten von $\mathcal{O}$, so daß diese Einheiten zusammen mit den Einheitswurzeln in $\mathcal{K}$ die komplette Einheitengruppe erzeugen. Die Laufzeit des Algorithmus ist $O((\log\Delta)^{3+\epsilon})$ für ein fixes r. Die Abhängigkeit der Laufzeit von r ist exponentiell. Die Erfolgswahrscheinlichkeit ist mindestens $2^{-20r^2-12r-2} r^{-4r^2}$.*

Beweis. Sei U die Einheitengruppe von $\mathcal{O}$ und $L = \{ \operatorname{Log} \alpha \mid \alpha \in U \}$. Setze

$$N = \left\lceil \max\{N(r,\Delta), 2^{2r+18} r^{(r+5)/2}\} \right\rceil .$$

Wir wenden den Algorithmus HSP-MD mit $q = 3rN$ auf die Funktion f_N an. Die Annahmen von Satz 6.1.3 sind erfüllt, da $\det L < w\sqrt{\Delta}$, was aus der Formel für Klassenzahl, $\lambda_1(L) > 1/2n$ und $\nu(L) \le 4(2r)^r 2^{n+1} n^2 \sqrt{\Delta} (\log\Delta)^{n-1} (\log\log\Delta)^{n/2}$ (siehe Satz 2.5.26) folgt. Wegen Lemma 6.2.4 wissen wir, daß die Funktion f_N in mehr als der Hälfte der Punkte $\mathcal{Q}(\mathbf{0})$ periodisch ist, da $q \gg 3rN(r,\Delta)$.

Algorithmus HSP-MD berechnet eine Menge der Vektoren $\{\mathbf{b}_1, \ldots, \mathbf{b}_r\}$. Als nächstes finden wir alle Minima von L, welche nahe an $\mathbf{b}_i$ liegen. Diese Suche kann in klassischer Polynomzeit (mit fixem r) durchgeführt werden (siehe Satz 2.5.44). Mit den gefundenen Minima können wir feststellen, ob alle $\mathbf{b}_i$'s nahe an Punkten aus L liegen. Wenn das der Fall ist, dann können wir die dazu gehörenden Einheiten berechnen.

Die Erfolgswahrscheinlichkeit des Algorithmus folgt aus dem Satz 6.1.3. □

Es ist zu bemerken, daß Algorithmus UnitGroup nicht in der Lage ist festzustellen, ob die Menge der Einheiten die komplette Einheitengruppe oder nur eine Untergruppe der Einheitengruppe erzeugt.

Kapitel 7

Sicherheits- und Laufzeitvergleich klassischer Kryptoverfahren

In diesem Kapitel wird die Sicherheit mehrerer klassischer Kryptoverfahren untersucht. Mit klassischen Kryptoverfahren sind Verfahren gemeint, deren Sicherheit auf dem Faktorisierungsproblem, dem DL-Problem in einer endlichen abelschen Gruppe sowie dem Hauptidealproblem in einem reell-quadratischen Zahlkörper basiert. Es wird ein neuer Begriff der Sicherheitsäquivalenz von Kryptoverfahren definiert. Darauf aufbauend werden die Laufzeiten und die Schlüsselgrößen von Kryptoverfahren bezüglich des neuen Äquivalenzbegriffs verglichen.

Klassisch gelten Kryptoverfahren als sicherheitsäquivalent, wenn die Laufzeiten der jeweils besten Angriffe darauf gleich sind [LV01]. Diese Laufzeiten sind entweder exponentiell, z.B. bei Kryptosystemen, deren Sicherheit auf dem DL-Problem in der Punktegruppe einer elliptischen Kurve basiert, oder subexponentiell, z.B. bei RSA oder Kryptoverfahren, deren Sicherheit auf dem DL-Problem in $\mathbb{Z}_p^\times$ oder Cl_Δ basiert. Mit Hilfe eines Quantencomputer können diese Kryptoverfahren jedoch in Polynomzeit gebrochen werden, so daß die Definition der Sicherheit über die Laufzeit nicht mehr sinnvoll ist.

Der weltweit erste Quantencomputer mit 2 Qubit wurde 1998 realisiert [CVZ+98]. Im Jahr 2000 wurde Shors Algorithmus erstmalig auf einem 5-Qubit-Quantencomputer implementiert [VSB+00]. Im Dezember 2001 bauten IBM-Forscher einen auf der NMR-Technologie basierenden Quantencomputer [VSB+01] und faktorisierten darauf mit Shors Algorithmus die Zahl 15. Der Quantencomputer bestand dabei aus 7 Qubits. Im Jahr 2005 wurde ein Quantenspeicher bestehend aus 8 Qubits implementiert [HHR+05].

Bisherige Forschungsergebnisse aus der Experementalphysik zeigen also, daß das Bauen von Quantencomputer sehr schwierig ist. Man kann also annehmen, daß die Größe der Quantencomputer auch in Zukunft nur langsam wachsen wird. Deshalb wird in dieser Arbeit ein neuer Sicherheitsbegriff definiert: die *Anzahl der Qubits*.

Satz 7.0.7 *Wir nennen zwei Kryptoverfahren* sicherheitsäquivalent*, wenn zum Lösen des zugrundeliegenden Problems die gleiche Anzahl der Qubits erforderlich ist.*

In den vorhergehenden Kapiteln ermittelten wir die Anzahl der Qubits zum Lösen des DL-Problems in der Klassengruppe quadratischer Zahlkörper sowie die Anzahl der Qubits zum

Lösen des Hauptidealproblems in reell quadratischen Zahlkörpern. Aus [Bea02] entnehmen wir die Anzahl der Qubits, die nötig sind, um eine Zahl zu faktorisieren. Die gleiche Anzahl der Qubits ist auch ausreichend um das DL-Problem in $\mathbb{Z}_p^\times$ zu lösen. Aus [PZ03] verwenden wir die Anzahl der Qubits zum Lösen des DL-Problem auf elliptischen Kurven über $\mathbb{F}_p$. Basierend auf diesen Ergebnissen untersuchen wir welche Kryptosysteme bei gegebener Größe der Quantencomputer die schnellsten sind.

Wir vergleichen die folgenden Kryptoverfahren: RSA, DSA und IES basierend auf $\mathbb{Z}_p^\times$, elliptischen Kurven über $\mathbb{F}_p$ und der Klassengruppe imaginär-quadratischer Zahlkörper sowie das Kryptosystem von Jacobson, Schneidler und Williams [JSW06] in reell-quadratischen Zahlkörpern. Auf das letzte Kryptosystem wird am Ende des Kapitels eingegangen.

7.1 Wahl eines Kryptosystems

Eine Analyse der Quantenalgorithmen zur Bestimmung von diskreten Logarithmen in endlichen abelschen Gruppen zeigt, daß die Anzahl der benötigten Qubits nicht vom Logarithmus abhängt. Diese Anzahl hängt nur von der Länge der Darstellung eines Gruppenelements und der Anzahl der temporären Qubits, die für die Ausführung einer Gruppenoperation nötig sind. Auf der anderen Seite ist die Laufzeit der Kryptoverfahren, die auf dem DL-Problem basieren, vom Logarithmus abhängig. Aus diesem Grund müssen schnelle und Quantencomputersichere Kryptosysteme, deren Sicherheit auf Problemen in endlichen abelschen Gruppen basieren, die folgende Eigenschaften erfüllen:

1. Die Länge der Darstellung der Gruppenelemente muß groß sein,
2. Die Qubit-Komplexität der Gruppenoperation muß groß sein,
3. Die Anzahl der Gruppenoperation für Ver-/Entschlüsselung bzw. Signatur/Verifikation muß klein sein.[1]

Das RSA-Kryptosystem erfüllt nicht die obengenannten Bedingungen. Insbesondere erfüllt es nicht die dritte Bedingung, da die Anzahl der Exponentiationen beim Entschlüsseln sehr groß ist. Analog verhält es sich mit dem ElGamal-Verfahren: Auch hier ist die Anzahl der Exponentiationen sehr groß. Beispiele für Kryptoverfahren, die die obengenannten Bedingungen erfüllen sind: DSA und IES. Bei diesen Kryptosystemen läßt sich die zugrundeliegende Gruppe und der diskrete Logarithmus unabhängig voneinander wählen.

7.2 Anzahl der Qubits

Wir fassen die Qubit-Komplexität in der folgenden Tabelle zusammen.

[1] Bei diesem Punkt müssen auch klassische Angriffe berücksichtigt werden.

Problem	Anzahl der Qubits zum Lösen (ca.)
Faktorisieren n	$2\log_2 n$
DL in $\mathbb{Z}_p^\times$	$2\log_2 p$
DL auf elliptischen Kurven über $\mathbb{Z}_p^\times$	$4\log_2 p + 8(\log_2 p)^{1/2}$
DL in IQ-Zahlkörpern mit Diskriminante Δ	$7.5\log_2\lvert\Delta\rvert + \log_2\log_2\lvert\Delta\rvert$
Hauptidealproblem in RQ-Zahlkörpern mit Diskriminante Δ	$13.5\log_2\Delta + 25\log_2^2(\log_2\Delta)$

Tabelle 7.1: Qubit-Komplexität von Algorithmen zur Lösung einiger zahlentheoretischer Probleme

In der ersten Zeile verwenden wir das Ergebnis aus [Bea02]. Die Qubit-Anzahl in der dritten Zeile kann erreicht werden, wenn im Algorithmus aus [PZ03] die Quantenaddition aus [Dra00] verwendet wird.

7.3 Vergleich der Laufzeiten

Basierend auf dem letzten Abschnitt führten wir verschiedene Tests durch, um die Laufzeiten der Kryptoverfahren zu messen, die sicherheitsäquivalent sind. Für das RSA-Kryptosystem wurden die Laufzeiten für das ver- und entschlüsseln gemessen (zum Verschlüsseln wurde der Exponent 65 verwendet). Für DSA und IES ermittelten wir die Zeit, die nötig ist, um ein zufälliges Gruppenelement zu potenzieren: Bei diesen Kryptosystemen ist das Potenzieren die zeitaufwendigste Operation. Als Potenz verwendeten wir eine 160-Bit-Zahl. Die Zahl 160 wurde aufgrund der Heuristiken von Lenstra und Verheul gewählt [LV01], um die klassische Sicherheit eines Kryptoverfahren für die nächsten zwanzig Jahre zu garantieren. Laut [LV01] wächst diese Zahl sehr langsam. Deshalb lassen sich diese Ergebnisse ohne große Änderungen auch dann verwenden, wenn die Sicherheit für längere Zeiträume erwünscht ist.

Die Laufzeiten wurden auf einem Pentium IV mit 1,73 GHz getestet. Für die Berechnungen verwendeten wir die folgenden Bibliotheken:

- GMP [GMP] für RSA und $\mathbb{F}_p$,
- GMP und LiDIA [LiD] für elliptische Kurven,
- GMP und libiq [Ham] für imaginär-quadratische Zahlkörper,
- GMP und Algorithmen aus [Saw04] und [SJ] für reell-quadratische Zahlkörper.

In der folgenden Tabelle präsentieren wir die mittleren Laufzeiten für die obengenannten Operationen. Die mit $(*)$ gekennzeichnet Zeiten stammen von Sicherheitsparametern, die so gewählt sind, damit sie heute den klassischen Angriffen widerstehen. Der Aufwand für solche Angriffe sowie die Sicherheitsparameter können [Ham02] und [Vol03] entnommen werden.

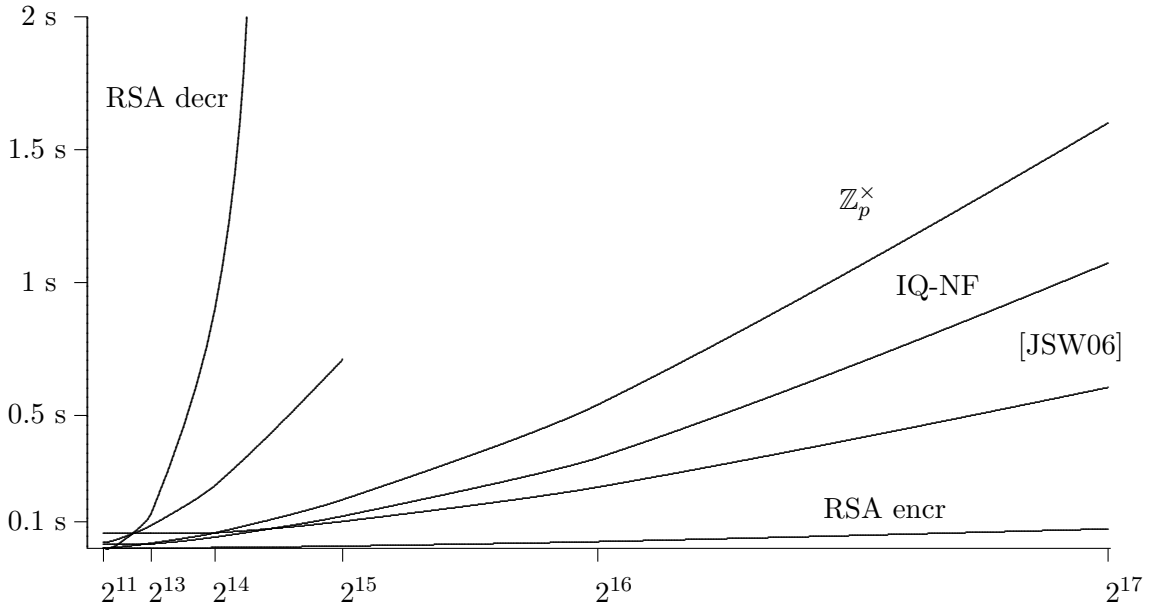

Abbildung 7.1: Laufzeitenvergleich. X-Koordinate: Anzahl der Qubits (Größe des Quantencomputers). Y-Koordinate: Mittlere Laufzeit für die RSA-Ver-/Entschlüsselung, sowie für die Potenzierung mit einer 160-Bit-Zahl. Die zugrunde liegende Gruppen sind gewählt, um dem Angriff eines Quantumcomputers gegebener Größe zu widerstehen.

Anzahl der Qubits (×1024)	2	4	8	16	32	64	128
RSA-Verschlüsselung (ms)	0.08	0.29	0.9	2.7	8.4	25	73
RSA-Entschlüsselung	2.5 ms	18 ms	130 ms	0.9 s	5.5 s	34 s	200 s
Mult auf ell. Kurven (ms)	23	36	86	236	710		
Exp in $\mathbb{Z}_p^\times$ (ms)	1.45	5.3	19	58	183	540	1600
Exp in IQ-NF (ms)	15(∗)	15(∗)	16.4	42	121	340	1073
Exp in RQ-NF (ms)	56(∗)	56(∗)	56(∗)	58	101	230	605

Tabelle 7.2: Laufzeitvergleich.

(∗) bedeutet in der Tabelle, daß die Zeiten den Schlüsselgrößen entsprechen, welche für klassische Sicherheit nötig sind.

In Abbildung 7.1 sind die Ergebnisse graphisch dargestellt. Diese Ergebnisse führen zu den folgenden Schlußfolgerungen. Das RSA-Kryptosystem wird sehr langsam, wenn Quantencomputer mit mehr als 8.000 Qubits gebaut werden. Kryptosysteme, die auf elliptischen Kurven basieren, verlieren ihre Geschwindigkeitsvorteile gegenüber DSA/IES in $\mathbb{Z}_p^\times$ und in der Klassengruppe quadratischer Zahlkörper sobald Quantencomputer mit 1.000 Qubits gebaut werden. Auf der anderen Seite bleiben DSA bzw. IES in $\mathbb{Z}_p^\times$ und in der Klassengruppe quadratischer Zahlkörper relativ schnell. Solange nicht Quantencomputer mehr als ca. 7.000 Qubits existieren, sind die Berechnungen in $\mathbb{Z}_p^\times$ schneller. Liegt die Größe der Quantencomputer zwischen ca. 7.000 und ca 25.000 Qubits, werden die Berechnungen in der Klassengruppe imaginär-quadratischer Zahlkörper schneller Wenn noch größere Quantencomputer gebaut werden, dann ist das Kryptosystem aus [JSW06] das schnellste. Deshalb sollte in diesem

Fall DSA bzw. IES in der Klassengruppe quadratischer Zahlkörper verwendet werden. Einen weiteren Grund für die Klassengruppe, werden wir in dem folgenden Abschnitt sehen.

7.4 Größe der Schlüssel

Neben der Laufzeit spielt auch die Größe des privaten bzw. öffentlichen Schlüssels eine wichtige Rolle. Es sind Kryptoverfahren bekannt, deren Laufzeiten sehr kurz sind, die jedoch wegen des langen Schlüssels praktisch nicht verwendet werden (z.B. McEliece [McE78]). Basierend auf der Tabelle auf der Seite 125 errechnen wir in diesem Abschnitt die relevanten Schlüsselgrößen für die schnellsten Kryptoverfahren, für DSA und IES in $\mathbb{Z}_p^\times$ und der Klassengruppe eines imaginär-quadratischen Zahlkörpers.

Bei DSA- bzw. IES-Verfahren in $\mathbb{Z}_p^\times$ besteht der Schlüssel aus einem Tripel (g, h, p), wobei p eine Primzahl, $g \in \mathbb{Z}_p^\times$ und h eine Potenz von g modulo p ist. Es ist möglich, für g eine kleine Zahl zu wählen, so daß die Größe von g eine kleine Konstante ist. Deshalb läßt sich ein öffentlicher Schlüssel mit ca. $2\,\mathrm{size}(p)$ Bits darstellen.

Der Schlüssel eines IQ-Kryptoverfahrens kann aus dem Tripel (a, b, c) bestehen, wobei (a, b, c) eine reduzierte Form der Diskriminante Δ ist. Aus $|b| \leq a \leq \sqrt{|\Delta|/3}$ und $4ac = |\Delta| + b^2 \leq |\Delta| + |\Delta|/3$ folgt, daß ein Schlüssel mit ca. $1.5\,\mathrm{size}(\Delta)$ Bits dargestellt werden kann.

Analog kann auch der Schlüssel im reell-quadratischen Fall durch eine reduzierte Form dargestellt werden. In diesem Fall gilt $0 < b < \sqrt{\Delta}$ und $|a| + |c| \leq \sqrt{\Delta}$. Daraus folgt, daß für einen Schlüssel ca. $1.5\,\mathrm{size}(\Delta)$ Bits ausreichend sind.

Anzahl der Qubits ($\times 1024$)	2	4	8	16	32	64	128
Schlüsselgröße in $\mathbb{Z}_p^\times$ ($\times 1024$)	2	4	8	16	32	64	128
Schlüsselgröße in $IQ - NF$ ($\times 1024$)	1.5 ($*$)	1.5 ($*$)	1.6	3.2	6.4	12.8	25.6
Schlüsselgröße in $RQ - NF$ ($\times 1024$)	1.5 ($*$)	1.5 ($*$)	1.5 ($*$)	1.5	3.3	6.8	14.1

Tabelle 7.3: Schlüsselgrößenvergleich

($*$) bedeutet in der Tabelle, daß die Schlüsselgrößen so gewählt sind, daß die Kryptosysteme heutigen klassischen Angriffen widerstehen.

Wir sehen, daß die Schlüssel von Kryptosystemen, die auf der Klassengruppe imaginär-quadratischer Zahlkörper basieren, immer kürzer als die von $\mathbb{Z}_p^\times$-Kryptosystemen sind. Die kürzesten Schlüssel haben jedoch die Kryptosysteme in reell-quadratischen Zahlkörpern.

7.5 Kryptosystem aus [JSW06]

In [JSW06] wurde ein neues Kryptoverfahren in reell-quadratischen Zahlkörpern präsentiert. Dieses Kryptosystem ist eine Adaption des Diffie-Hellmann-Protokolls auf die Infrastruktur reell-quadratischer Zahlkörper.

Durch die Verwendung des NUCOMP-Algorithmus zur Idealmultiplikation und einer neuen (f, p)-Darstellung (siehe [JSW06] für weitere Details) wurde das Kryptosystem entscheidend beschleunigt. Der Vorteil von NUCOMP ist, daß bei der Multiplikation von zwei reduzierten Idealen die meisten Zwischenergebnisse in der Größenordnung $O(\sqrt{\Delta})$ und nicht mehr $O(\Delta)$, wie bei der üblichen Idealmultiplikation (siehe Definition 2.5.46) sind. Der Vorteil der (f, p)-Darstellung ist, daß sie die Distanzberechnung ersetzt, welche sonst die zeitaufwendigste Operation ist. In [Saw04] wurde der Algorithmus nochmal verbessert und implementiert. Diese Implementierung verwendeten wir für die Laufzeittests.

Kapitel 8

Zur Berechnung der Struktur endlicher abelscher Gruppen

In diesem Kapitel wird ein Algorithmus für klassische Computer zur Berechnung der Struktur einer endlichen abelschen Gruppe $\mathcal{G}$ präsentiert. Die exakte Definition des Strukturproblems ist auf der Seite 7 gegeben. Als Eingabe bekommt der Algorithmus ein Erzeugendensystem $\mathcal{M}$ von $\mathcal{G}$.

Es wird angenommen, daß die folgenden Operationen in der Gruppe $\mathcal{G}$ durchgeführt werden können:

- Multiplikation von zwei Gruppenelementen,
- Berechung des Inverses eines Gruppenelements und
- Gleichheitstest, d.h. ein Test, welcher feststellt, ob zwei Gruppenelemente gleich sind oder nicht.

Der bisher beste Algorithmus wurde in [BJT97] vorgestellt. Seine Laufzeit hängt exponentiell von der Länge des Erzeugendensystems ab ($O(2^{|\mathcal{M}|}\sqrt{|\mathcal{G}|})$). Dagegen ist die Laufzeit des in dieser Arbeit entwickelten Algorithmus $O(|\mathcal{M}|\sqrt{|\mathcal{G}|})$, d.h. sie hängt nur linear von der Länge des Erzeugendensystems ab.

Das Kapitel ist wie folgt aufgebaut. Als erstes beschreiben wir den Algorithmus von Terr zur Berechnung der Ordnung eines Gruppenelements. Im zweiten Abschnitt präsentieren wir den Algorithmus zur Berechnung der Struktur einer endlichen abelschen Gruppe.

8.1 Berechnung der Ordnung eines Gruppenelements

In diesem Abschnitt beschreiben wir einen Algorithmus zur Berechnung der Ordnung eines Elements g einer endlichen abelschen Gruppe G. Dieser Algorithmus ist ein Spezialfall des Algorithmus von Terr [Ter00]. Er beruht auf dem folgenden Lemma.

Lemma 8.1.1 *Sei $g \in \mathcal{G}$. Dann gibt es ein $e \in \mathbb{N}$ und $f \in \{0, \ldots, e-1\}$ mit $g^{e(e+1)/2} = g^f$. Ist e minimal gewählt, dann gilt: $e(e-1)/2 < \text{order}(g) \leq e(e+1)/2$, f ist eindeutig bestimmt und $\text{order}(g) = e(e+1)/2 - f$.*

Beweis. Sei $e \in \mathbb{N}$, so daß $e(e-1)/2 < \text{order}(\mathsf{g}) \leq e(e+1)/2$. Ein solches e existiert, da $e(e-1)/2 + e = e(e+1)/2$. Sei nun $f = e(e+1)/2 - \text{order}(\mathsf{g})$, dann gilt $f \in \{0, \ldots, e-1\}$. Aus $\mathsf{g}^{e(e+1)/2-f} = \mathsf{g}^{\text{order}(\mathsf{g})} = 1$ folgt schließlich $\mathsf{g}^{e(e+1)/2} = \mathsf{g}^f$. Dies beweist die Existenz von e und f.

Als nächstes beweisen wir die Minimalität von e. Seien $e' \in \mathbb{N}$, $f' \in \{0, \ldots, e'-1\}$ so, daß $\mathsf{g}^{e'(e'+1)/2-f'} = 1$. Dann gilt $e'(e'+1)/2 \geq e'(e'+1)/2 - f' \geq \text{order}(\mathsf{g}) = e(e+1)/2 - f > e(e-1)/2$. Die Zahlen e und e' sind ganze Zahlen, daraus folgt $e'(e'-1)/2 \geq e(e-1)/2$ und deshalb $e' \geq e$. □

Für $e = 1, 2, \ldots$ berechnet Terrs Algorithmus die Menge

$$\mathsf{babySet} = \{\, (\mathsf{g}^f, f) \mid 0 \leq f < e \,\}$$

und überprüft, ob es ein Paar $(\mathsf{g}^{e(e+1)/2}, f)$, mit einem $f \in \mathbb{N}$, in babySet gibt. Wegen Lemma 8.1.1 existiert ein solches Paar. Für das kleinste $e \in \mathbb{N}$, für das ein solches Paar gefunden wird, gilt $\text{order}(\mathsf{g}) = e(e+1)/2 - f$. Wir verwenden im folgenden Algorithmus die Bezeichnungen:

$$\mathsf{babyElement} = \mathsf{g}^e \quad \text{und} \quad \mathsf{giantElement} = \mathsf{g}^{e(e+1)/2}$$

Algorithm 49 ORDER(g)

Input: Ein Gruppenelement g.
Output: Die Ordnung n von g.

```
babySet ← {(1, 0)}.
e ← 1
babyElement ← g
giantElement ← g
loop
  if babySet contains a pair (giantElement, f) then return n = e(e + 1)/2 − f
  insert (babyElement, e) in babySet
  babyElement ← g · babyElement
  giantElement ← giantElement · babyElement
  e ← e + 1
```

Satz 8.1.2 *Sei $g \in \mathcal{G}$ die Eingabe des Algorithmus* ORDER. *Sei außerdem $n = \text{order}(g)$. Dann terminiert* ORDER *stets und gibt n aus. Der Algorithmus durchläuft höchstens $\sqrt{2n}+1/2$ Iterationen und führt dabei höchstens $2\sqrt{2n}-1$ Gruppenoperationen und $\sqrt{2n}+1/2$ suchende Tabellenzugriffe aus. Während der Ausführung enthält die Menge babySet höchstens $\sqrt{2n}+1/2$ Elemente aus G.*

Beweis. Aus Lemma 8.1.1 folgt, daß der Algorithmus ORDER terminiert, dabei $n = \text{order}(\mathsf{g})$ ausgibt und daß $e(e-1)/2 < n \leq e(e+1)/2$ gilt. Da e und n ganze Zahlen sind, folgern wir,

daß $(e-1/2)^2 = e(e-1)+1/4 < 2n$ gilt und deshalb e, die Anzahl der Iterationen, kleiner als $\sqrt{2n}+1/2$ ist.

In den ersten $(e-1)$ Iterationen der Schleife werden zwei Gruppenoperationen, ein suchender und ein schreibender Tabellenzugriff ausgeführt. In der letzten Iteration wird nur einmal suchend auf die Tabelle zugegriffen. Daraus folgen die restlichen Behauptungen des Lemmas. □

8.2 Berechnung der Gruppenstruktur

Aufbauend auf dem Algorithmus aus dem letzten Abschnitt, werden wir nun einen Algorithmus zur Berechnung der Struktur einer endlichen abelschen Gruppe vorstellen.

Sei

$$M = (\mathsf{g}_1, \ldots, \mathsf{g}_l)$$

ein Erzeugendensystem für $\mathcal{G}$. Für einen Vektor $\mathbf{q} = (q_1, \ldots, q_l) \in \mathbb{Z}^l$ schreiben wir

$$M^{\mathbf{q}} = \prod_{j=1}^{l} \mathsf{g}_j^{q_j}.$$

Eine *Relation* für M ist ein Vektor $\mathbf{q}$ mit $M^{\mathbf{q}} = 1$. Die Menge aller Relationen für M ist ein Gitter in $\mathbb{Z}^l$, wir bezeichnen dieses Gitter mit $L(M)$. Seine Dimension ist l, denn das Gitter ist der Kern eines surjektiven Homomorphismus

$$\mathbb{Z}^l \longrightarrow G, \quad \mathbf{q} \mapsto M^{\mathbf{q}}. \tag{8.1}$$

Sei außerdem $\mathsf{U} = (\mathbf{u}_1, \ldots, \mathbf{u}_m) \in \mathbb{Z}^{l,m}$ eine Matrix, dann schreiben wir

$$M^{\mathsf{U}} = (M^{\mathbf{u}_1}, \ldots, M^{\mathbf{u}_m}).$$

Unser Algorithmus basiert auf dem folgenden Lemma.

Lemma 8.2.1 *Sei* $\mathsf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_l) \in \mathbb{Z}^{(l,l)}$ *eine Basis des Relationengitters* $L(M)$. *Seien außerdem* $\mathsf{U}', \mathsf{V} \in GL(l, \mathbb{Z})$ *so, daß* $\mathsf{U}'\mathsf{D} = \mathsf{B}\mathsf{V}$, *wobei* $\mathsf{D} = \mathrm{diag}(n_1, \ldots, n_k, 1, \ldots, 1)$ *mit* $n_k > 1$ *eine Matrix in Smith-Normalform ist. Setze* $\mathsf{U} \equiv \mathsf{U}' \pmod{|\mathcal{G}|}$ *und* $M^{\mathsf{U}} = (\mathsf{h}_1, \ldots, \mathsf{h}_k, \ldots, \mathsf{h}_l)$, *dann gilt:*

1. *Die Ordnung von* $\mathcal{G}$ *ist* $|\det(\mathsf{B})|$,
2. *die Invarianten von* $\mathcal{G}$ *sind* $n_1, \ldots, n_k$,
3. *die Ordnung von* h_i *ist* n_i, $1 \le i \le k$, *und*

$$\mathcal{G} = \langle \mathsf{h}_1 \rangle \times \cdots \times \langle \mathsf{h}_k \rangle. \tag{8.2}$$

Beweis. Die Determinante von B ist der Index des Kerns des Homomorphismus aus (8.1) in $\mathbb{Z}^l$. Dieser Index ist die Ordnung von $\mathcal{G}$.

Als nächstes beweisen wir die zweite und die dritte Aussage des Lemmas. Wir zeigen zurerst, daß M^{U} ein Erzeugendensystem für $\mathcal{G}$ ist. Einerseits gilt $(M^{\mathsf{U}})^{\mathbf{v}} \in \mathcal{G}$ für jedes $\mathbf{v} \in \mathbb{Z}^l$. Andererseits gibt es für ein beliebiges $\mathsf{g} \in \mathcal{G}$ ein $\mathbf{v} \in \mathbb{Z}^l$ so, daß $\mathsf{g} = M^{\mathbf{v}}$. Da $\det \mathsf{U} = 1$ ist, gilt $\mathrm{ggT}(\det \mathsf{U}, |\mathcal{G}|) = 1$, woraus die Existenz einer Matrix $\mathsf{W} \in \mathbb{Z}^{(l,l)}$ folgt, für die $\mathsf{UW} \equiv \mathsf{I}_l \pmod{|\mathcal{G}|}$ gilt, wobei I_l die Einheitsmatrix der Dimension l ist. Wir setzen $\mathbf{w} = \mathsf{W}\mathbf{v}$ und folgern

$$(M^{\mathsf{U}})^{\mathbf{w}} = (M^{\mathsf{U}})^{\mathsf{W}\mathbf{v}} = M^{\mathsf{UW}\mathbf{v}} = M^{\mathbf{v}} = \mathsf{g}.$$

Damit ist gezeigt, daß M^{U} ein Erzeugendensystem für $\mathcal{G}$ ist.

Als nächstes zeigen wir, daß die Spalten von D eine Basis des Relationengitters $L(M^{\mathsf{U}})$ darstellen. Aus $(M^{\mathsf{U}})^{\mathsf{D}} = M^{\mathsf{UD}} = M^{\mathsf{U'D}} = M^{\mathsf{BV}}$ folgt, daß die Spalten von D Relationen für M^{U} sind. Sei $\mathbf{v}$ eine beliebige Relation für M^{U}. Wegen $(M^{\mathsf{U}})^{\mathbf{v}} = M^{\mathsf{U'}\mathbf{v}}$ ist $\mathsf{U'}\mathbf{v}$ eine Relation für M. Da BV eine Basis von $L(M)$ ist, gibt es ein $\mathbf{x} \in \mathbb{Z}^l$ mit $\mathsf{U'}\mathbf{v} = \mathsf{BV}\mathbf{x} = \mathsf{U'D}\mathbf{x}$, woraus $\mathbf{v} = \mathsf{D}\mathbf{x}$ folgt. Da $\mathbf{v}$ eine beliebige Relation für M^{U} ist, ist D eine Basis des Relationengitters $L(M^{\mathsf{U}})$.

Da $\mathsf{D} = \mathrm{diag}(n_1, \ldots, n_k, 1, \ldots, 1)$ eine Diagonalmatrix ist, ist die Ordnung des Elements h_i gleich n_i, $1 \le i \le l$. Außerdem gilt $\mathsf{h}_{k+1} = \ldots = \mathsf{h}_l = 1$. Daraus folgt, daß $(\mathsf{h}_1, \ldots, \mathsf{h}_k)$ ein Erzeugendensystem für $\mathcal{G}$ ist. Die Gleichung (8.2) folgt aus der Tatsache, daß D eine Basis von $L(M^{\mathsf{U}})$ ist. Falls nämlich $(\mathsf{h}_1, \ldots, \mathsf{h}_k)^{\mathbf{y}} = 1$ für ein $\mathbf{y} = (y_1, \ldots, y_k) \in \mathbb{Z}^k$ gilt, so ist $y_i \equiv 0 \pmod{n_i}$, $1 \le i \le k$. Da D die Smith-Normalform von B ist, folgt schließlich, daß $n_1, \ldots, n_k$ Invarianten von $\mathcal{G}$ sind. □

Aufgrund des Lemmas können wir die Struktur von $\mathcal{G}$ wie folgt berechnen. Zuerst bestimmen wir eine Basis B des Relationengitters $L(M)$. Mit Standardtechniken (siehe [HM91], [Coh00]) können wir die Smith-Normalform von B berechnen, und erhalten dadurch Matrizen D und U wie in Lemma 8.2.1. Schließlich können die Invarianten von $\mathcal{G}$ und die Repräsentation von $\mathcal{G}$ als Produkt zyklischer Gruppen, deren Ordnungen die Invarianten sind, wie im Lemma beschrieben berechnet werden.

8.2.1 Berechnung des Relationengitters

Nun beschreiben wir, wie eine spezielle Basis $\mathsf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_l)$ des Relationengitters $L(M)$ berechnet werden kann, nämlich die Basis B in hermiter Normalform. Wir schreiben

$$\mathbf{b}_j = (b_{1,j}, \ldots, b_{l,j}), \quad 1 \le j \le l,$$

und es gilt $b_{j,j} \ge 1$ für $1 \le j \le l$, $b_{i,j} = 0$ für $0 \le j < i \le l$ und $0 \le b_{i,j} < b_{j,j}$ für $1 \le i < j \le l$.

Sei nun $j \in \{1, \ldots, l\}$. Wir nehmen an, daß wir bereits die Basisvektoren $\mathbf{b}_1, \ldots \mathbf{b}_{j-1}$ berechnet haben. Dann lässt sich $\mathbf{b}_j$ wie folgt bestimmen. Sei $\mathbf{e}_i$ der i-te Einheitsvektor, d.h.

$$\mathbf{e}_i = (\underbrace{0, \ldots, 0}_{i-1}, 1, \underbrace{0, \ldots, 0}_{l-i}) \in \mathbb{Z}^l \quad , 1 \le i \le l.$$

Bezeichne mit $\mathcal{H}$ die Untergruppe von $\mathcal{G}$, die von $\mathsf{g}_1, \ldots, \mathsf{g}_{j-1}$ erzeugt wird, d.h.

$$\mathcal{H} = \{ \prod_{i=1}^{j-1} \mathsf{g}_i^{x_i} \mid 0 \le x_i < b_{i,i}, 1 \le i < j \}. \tag{8.3}$$

Die Untergruppe $\mathcal{H}$ hängt von j ab. Aus Gründen der besseren Lesbarkeit verzichten wir jedoch darauf, den Index j explizit zu schreiben.

Der Eintrag $b_{j,j}$ der Matrix B ist die Ordnung der Nebenklasse $\mathsf{g}_j\mathcal{H}$ in der Faktorgruppe $\mathcal{G}/\mathcal{H}$. Um diesen Eintrag zu berechnen, verwenden wir den Algorithmus ORDER aus dem letzten Abschnitt. Wir suchen also die kleinste Zahl e, für die

$$\mathsf{g}_j^{e(e+1)/2} = \mathsf{g}_j^f \mathsf{h}, \tag{8.4}$$

für ein $f \in \{0, \ldots, e-1\}$ und ein $\mathsf{h} \in \mathcal{H}$ gilt. Um e zu finden, wurden im Algorithmus ORDER alle Gruppenelemente, die auf der rechten Seite der Gleichung (8.4) stehen, in babySet gespeichert. Da jedoch $\mathcal{H}$ genauso viele Elemente wie $\mathcal{G}$ enthalten kann, eignet sich diese Methode nicht, um die gewünschte Komplexität, $O(l(M)\sqrt{|\mathcal{G}|})$, zu erreichen. Stattdessen zerlegen wir $\mathcal{H}$ in zwei Teile. Wir zerlegen die Menge $\{1, \ldots, j-1\}$ in drei paarweise disjunkte Mengen

$$\{1, \ldots, j-1\} = \mathcal{I}_1 \cup \{m\} \cup \mathcal{I}_2 \tag{8.5}$$

und setzen

$$\mathcal{H}_1 = \{ (M^{-\mathbf{v}}, \mathbf{v}) \mid \mathbf{v} = \sum_{i \in \mathcal{I}_1} x_i \mathbf{e}_i,\ 0 \le x_i < b_{i,i},\ i \in \mathcal{I}_1 \} \tag{8.6}$$

und

$$\mathcal{H}_2 = \{ (M^{\mathbf{v}}, \mathbf{v}) \mid \mathbf{v} = \sum_{i \in I_2} x_i \mathbf{e}_i,\ 0 \le x_i < b_{i,i},\ i \in \mathcal{I}_2 \}. \tag{8.7}$$

Die Aufteilung in (8.5) ist so zu wählen, daß

$$|\mathcal{H}_i| \le \sqrt{|\mathcal{H}|}, \quad i = 1, 2. \tag{8.8}$$

Wir setzen

$$s = \left\lceil \sqrt{|\mathcal{H}|}/|\mathcal{H}_1| \right\rceil \tag{8.9}$$

und

$$t = \left\lceil \sqrt{|\mathcal{H}|}/|\mathcal{H}_2| \right\rceil. \tag{8.10}$$

Das folgende Lemma beschreibt den Zusammenhang zwischen $\mathcal{H}$, $\mathcal{H}_1$ und $\mathcal{H}_2$.

Lemma 8.2.2 *Für jedes $\mathsf{h} \in \mathcal{H}$ gibt es $(\mathsf{h}_1, \mathbf{v}_1) \in \mathcal{H}_1$ und $(\mathsf{h}_2, \mathbf{v}_2) \in \mathcal{H}_2$ so, daß $\mathsf{h} = \mathsf{h}_1^{-1} \mathsf{g}_m^{qs+r} \mathsf{h}_2$ mit $0 \le q < t$ und $0 \le r < s$.*

Beweis. Sei $\mathsf{h} \in \mathcal{H}$ beliebig. Wegen (8.3) und (8.5) gibt es $(\mathsf{h}_1, \mathbf{v}_1) \in \mathcal{H}_1$, $(\mathsf{h}_2, \mathbf{v}_2) \in \mathcal{H}_2$ und $n \in \{0, \ldots, b_{m,m} - 1\}$ so, daß

$$\mathsf{h} = \mathsf{h}_1^{-1} \mathsf{g}_m^n \mathsf{h}_2$$

Wir wählen $r, q \in \mathbb{Z}$ so, daß $n = qs + r$ mit $0 \le r < s$. Dann gilt $qs < b_{m,m}$ und es folgt $q < b_{m,m}/s \le b_{m,m}|\mathcal{H}_1|/\sqrt{|\mathcal{H}|} = |\mathcal{H}|/(|\mathcal{H}_2|\sqrt{|\mathcal{H}|}) \le t$. □

Basierend auf dem letzten Lemma können wir die Gleichung (8.4) folgendermaßen modifizieren

$$\mathsf{g}_j^{e(e+1)/2}\mathsf{h}_2\mathsf{g}_m^{qs} = \mathsf{g}_j^{f}\mathsf{h}_1\mathsf{g}_m^{-r}, \tag{8.11}$$

wobei $(\mathsf{h}_i, \mathbf{v}_i) \in \mathcal{H}_i$, $i = 1, 2$, $0 \le r < s$, $0 \le q < t$ und $0 \le f < e$. Um $\mathbf{b}_j$ zu bestimmen, suchen wir nun das kleinste e, für das die Gleichung (8.11) erfüllt ist. Wenn wir ein solches e gefunden haben, dann setzen wir $\mathbf{b}_j$ wie folgt zusammen

$$\mathbf{b}_j = \mathbf{v}_1 + \mathbf{v}_2 + (qs + r)\mathbf{e}_m + (e(e+1)/2 - f)\mathbf{e}_j. \tag{8.12}$$

Um eine Übereinstimmung wie in (8.11) zu finden, verwenden wir zwei Mengen. Die erste Menge ist

$$\mathsf{babySet} = \{\, (\mathsf{g}_j^{f}\mathsf{h}, \mathbf{v} - f\mathbf{e}_j) \mid (\mathsf{h}, \mathbf{v}) \in \mathsf{auxiliaryBabySet}, 0 \le f < e \,\}$$

mit

$$\mathsf{auxiliaryBabySet} = \{\, (\mathsf{h}_1\mathsf{g}_m^{-r}, \mathbf{v} + r\mathbf{e}_m) \mid (\mathsf{h}_1, \mathbf{v}) \in \mathcal{H}_1, 0 \le r < s \,\}$$

In babySet speichern wir die Elemente der rechten Seite von (8.11). Die zweite Menge ist

$$\mathsf{giantSet} = \{\, (\mathsf{h}_2\mathsf{g}_m^{qs}, \mathbf{v} + qs\mathbf{e}_m) \mid (\mathsf{h}_2, \mathbf{v}) \in \mathcal{H}_2, 0 \le q < t \,\}.$$

Wie im Algorithmus Order verwenden wir

$$\mathsf{babyElement} = \mathsf{g}_j^{e} \quad \text{und} \quad \mathsf{giantElement} = \mathsf{g}_j^{e(e+1)/2}.$$

In der e-ten Iteration multiplizieren wir giantElement mit jedem Element aus giantSet und prüfen, ob das Produkt in babySet liegt. Wenn das der Fall ist, dann können wir $\mathbf{b}_j$ wie in (8.12) ausrechnen. Andererfalls erhöhen wir e um eins, aktualisieren babySet, babyElement und giantElement und wiederholen die Prozedur.

Wenn $\mathbf{b}_j$ bestimmt wurde und $j = l$ ist, terminiert der Algorithmus. Ist dagegen $j < l$, so wird eine neue Partition (8.5) bestimmt und die Mengen $\mathcal{H}_1$, $\mathcal{H}_2$, auxiliaryBabySet und giantSet aktualisiert.

Wir präsentieren nun den Algorithmus.

Algorithm 50 HNFRelationBasis(M)

Input: Ein Erzeugendensystem $M = (\mathsf{g}_1, \ldots, \mathsf{g}_l)$ für $\mathcal{G}$
Output: Die Relationenbasis $\mathsf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_l)$ von $L(M)$ in hermiter Normalform

$\mathcal{H}_i \leftarrow \{(1, (0, \ldots, 0))\}$, $i = 1, 2$
$\mathcal{I}_i \leftarrow \emptyset$, $i = 1, 2$
$s \leftarrow 0$, $t \leftarrow 0$, $m \leftarrow 0$
auxiliaryBabySet $\leftarrow \mathcal{H}_1$, giantSet $\leftarrow \mathcal{H}_2$
for $j = 1, \ldots, l$ **do**
  $e \leftarrow 1$
  babySet $\leftarrow$ auxiliaryBabySet, babyElement $\leftarrow \mathsf{g}_j$, giantElement $\leftarrow \mathsf{g}_j$
  loop
    for all $(\mathsf{g}, \mathbf{v}) \in$ giantSet **do**
      if babySet contains a pair $(\mathsf{g} \cdot \text{giantElement}, \mathbf{w})$ **then**
        $\mathbf{b}_j \leftarrow \mathbf{v} + \mathbf{w} + (e(e+1)/2)\mathbf{e}_j$
        break
    babySet $\leftarrow$ babySet $\cup \{ (\mathsf{g} \cdot \text{babyElement}, \mathbf{v} - e\mathbf{e}_j) \mid (\mathsf{g}, \mathbf{v}) \in \text{auxiliaryBabySet} \}$
    $e \leftarrow e + 1$, babyElement $\leftarrow$ babyElement $\cdot\, \mathsf{g}_j$, giantElement $\leftarrow$ giantElement $\cdot$ babyElement
  if $j < l$ and $b_{j,j} > 1$ **then**
    if $b_{j,j} \prod_{i \in \mathcal{I}_1} b_{i,i} \leq \sqrt{\prod_{i=1}^{j} b_{i,i}}$ **then**
      $\mathcal{H}_1 \leftarrow \mathcal{H}_1 \cup \{ (\mathsf{g}_j^{-x}\mathsf{g}, \mathbf{v} + x\mathbf{e}_j) \mid (\mathsf{g}, \mathbf{v}) \in \mathcal{H}_1, 0 \leq x < b_{j,j} \}$
      $\mathcal{I}_1 \leftarrow \mathcal{I}_1 \cup \{j\}$
    else
      if $m > 0$ **then**
        $\mathcal{H}_2 \leftarrow \mathcal{H}_2 \cup \{ (\mathsf{g}_m^{x}\mathsf{g}, \mathbf{v} + x\mathbf{e}_m) \mid (\mathsf{g}, \mathbf{v}) \in \mathcal{H}_2, 0 \leq x < b_{m,m} \}$
        $\mathcal{I}_2 \leftarrow \mathcal{I}_2 \cup \{m\}$
      $m \leftarrow j$
    $s \leftarrow \lceil \sqrt{\prod_{i=1}^{j} b_{i,i}} / \prod_{i \in I_1} b_{i,i} \rceil$
    $t \leftarrow \lceil \sqrt{\prod_{i=1}^{j} b_{i,i}} / \prod_{i \in I_2} b_{i,i} \rceil$
    auxiliaryBabySet $\leftarrow \{ (\mathsf{h}_1\mathsf{g}_m^{-r}, \mathbf{v} + r\mathbf{e}_m) \mid (\mathsf{h}_1, \mathbf{v}) \in \mathcal{H}_1, 0 \leq r < s \}$
    giantSet $\leftarrow \{ (\mathsf{h}_2\mathsf{g}_m^{qs}, \mathbf{v} + qs\mathbf{e}_m) \mid (\mathsf{h}_2, \mathbf{v}) \in \mathcal{H}_2, 0 \leq q < t \}$
return $(\mathbf{b}_1, \ldots, \mathbf{b}_k)$

Lemma 8.2.3 *1. Sei $k \in \mathbb{N}$ und $a_1, \ldots, a_k \in \mathbb{R}_{\geq 1}$, dann gilt $\sum_{j=1}^{k} a_j \leq \prod_{j=1}^{k} a_j + (k-1)$*

2. Sei $k \in \mathbb{N}$ und $a_1, \ldots, a_k \in \mathbb{R}_{\geq 2}$, dann gilt $\sum_{j=1}^{k} \prod_{i=1}^{j} \sqrt{a_j} \leq (2 + \sqrt{2}) \prod_{j=1}^{k} \sqrt{a_j}$

Beweis. 1. Für alle $x, y \in \mathbb{R}_{\geq 1}$ gilt

$$x + y - xy - 1 = \underbrace{(x-1)}_{\geq 0}\underbrace{(1-y)}_{\leq 0} \leq 0.$$

Daraus folgt

$$x + y \leq xy + 1.$$

Wir beweisen die erste Aussage des Lemmas mit vollständige Induktion über k.

Für $k = 1$ ist die Aussage offensichtlich wahr.

Wir nehmen nun an, daß die Aussage für $k-1$ gilt, dann gilt auch

$$\sum_{j=1}^{k} a_j = \sum_{j=1}^{k-1} a_j + a_k \leq \underbrace{\prod_{j=1}^{k-1} a_j}_{\geq 1} + a_k + (k-2) \leq \prod_{j=1}^{k} a_j + (k-1).$$

2. Sei $B = \prod_{j=1}^{k} \sqrt{a_j}$, dann gilt

$$\sum_{j=1}^{k} \left(\prod_{i=1}^{j} \sqrt{a_i} \right) = B \sum_{j=1}^{k} \left(\prod_{i=j+1}^{k} \frac{1}{\sqrt{a_i}} \right) \leq B \sum_{j=1}^{k} \left(\prod_{i=j+1}^{k} \frac{1}{\sqrt{2}} \right) = B \sum_{j=1}^{k} \left(\frac{1}{\sqrt{2}} \right)^{k-j} =$$

$$B \sum_{j=0}^{k-1} \left(\frac{1}{\sqrt{2}} \right)^{j} = B \frac{1 - (1/\sqrt{2})^k}{1 - 1/\sqrt{2}} = B(2 + \sqrt{2} - \frac{2+\sqrt{2}}{\sqrt{2}^k}) \leq (2+\sqrt{2}) \prod_{j=1}^{k} \sqrt{a_j}$$

□

Satz 8.2.4 *Der Algorithmus* HNFRelationBasis *berechnet die HNF-Basis des Relationengitters von M und*

- *führt höchstens* $(48 + 9l - 6l(M))\sqrt{|\mathcal{G}|} + 2l(M) \log \sqrt{|\mathcal{G}|}$ *Gruppenoperationen in $\mathcal{G}$ aus und*
- *sucht höchstens* $4(2 + \sqrt{2} + l - l(M))\sqrt{|\mathcal{G}|}$ *mal nach einem Tupel in einer Menge.*

Der Algorithmus verwendet

- *zwei Tabellen mit höchstens* $\sqrt{|\mathcal{G}|}$,
- *zwei Tabellen mit höchstens* $2\sqrt{|\mathcal{G}|}$ *und*
- *eine Tabelle mit höchstens* $4\sqrt{|\mathcal{G}|}$

Tupel $(\mathbf{g}, \mathbf{q}) \in \mathcal{G} \times \{0, \ldots, \left\lfloor \sqrt{|\mathcal{G}|} \right\rfloor\}^l$.

Beweis. Als erstes zeigen wir, warum die Ausgabe des Algorithmus HNFRelationBasis die HNF-Basis des Relationengitters von L des Erzeugendensystem M ist. Aus den Definitionen (8.6), (8.7) und (8.12) sowie aus der Gleichung (8.11) folgt, daß $\mathbf{b}_j$, $1 \leq j \leq l$, eine Relation ist. Durch der Konstruktion bilden die b_j's eine Matrix B in hermiter Normalform. Wegen der Wahl von $b_{j,j}$'s gilt $\det \mathsf{B} = |\mathcal{G}| = \det L$. Daraus folgt das B die HNF-Basis von L ist.

Als nächstes bestimmen wir die Größen der Mengen $\mathcal{H}_1$, $\mathcal{H}_2$, babySet, auxiliaryBabySet und giantSet. Wir bezeichnen mit $\mathcal{H}_1(j)$ bzw. $\mathcal{H}_2(j)$ die Menge $\mathcal{H}_1$ bzw. $\mathcal{H}_2$ am Anfang der j-ten Iteration. Sei außerdem $\mathcal{G}_j = \langle \mathsf{g}_1, \ldots, \mathsf{g}_j \rangle$, $j = 0, \ldots, l$, wobei $\mathcal{G}_0$ die triviale Gruppe ist, die nur aus dem neutralen Element besteht.

Ist $j = 1$, dann gilt $|\mathcal{H}_1(1)| = |\mathcal{H}_2(1)| = \sqrt{|\mathcal{G}_0|} \leq \sqrt{|\mathcal{G}|}$. Nun nehmen wir an, daß die Bedingung $|\mathcal{H}_i(j-1)| \leq |\mathcal{G}_{j-1}|$, mit $i = 1, 2$ und $j = 2, \ldots, l-1$, erfüllt ist und zeigen, daß dann $|\mathcal{H}_i(j)| \leq |\mathcal{G}_j|$, $i = 1, 2$ gilt. Wir unterscheiden drei Fälle:

- Ist $b_{j,j} \prod_{i \in \mathcal{I}_1} b_{i,i} \leq \sqrt{\prod_{i=1}^{j} b_{i,i}}$, dann gilt

$$|\mathcal{H}_1(j)| = b_{j,j}|\mathcal{H}_1(j-1)| = b_{j,j} \prod_{i \in \mathcal{I}_1} b_{i,i} \leq \sqrt{\prod_{i=1}^{j} b_{i,i}} \leq \sqrt{|\mathcal{G}_j|} \leq \sqrt{|\mathcal{G}|}$$

und

$$|\mathcal{H}_2(j)| = |\mathcal{H}_2(j-1)| \leq \sqrt{|\mathcal{G}_j|} \leq \sqrt{|\mathcal{G}|},$$

- ist $b_{j,j} \prod_{i \in \mathcal{I}_1} b_{i,i} \geq \sqrt{\prod_{i=1}^{j} b_{i,i}}$ und $b_{m,m} \leq b_{j,j}$, dann gilt

$$|\mathcal{H}_1(j)| = |\mathcal{H}_1(j-1)| \leq \sqrt{|\mathcal{G}_j|} \leq \sqrt{|\mathcal{G}|}$$

und

$$|\mathcal{H}_2(j)| = b_{m,m}|\mathcal{H}_2(j-1)| \leq \sqrt{b_{m,m}|\mathcal{G}_{j-1}|} \leq \sqrt{b_{j,j}|\mathcal{G}_{j-1}|} = \sqrt{|\mathcal{G}_j|} \leq \sqrt{|\mathcal{G}|},$$

- ist schließlich $b_{j,j} \prod_{i \in \mathcal{I}_1} b_{i,i} \geq \sqrt{\prod_{i=1}^{j} b_{i,i}}$ und $b_{m,m} > b_{j,j}$, dann gilt

$$|\mathcal{H}_1(j)| = |\mathcal{H}_1(j-1)| \leq \sqrt{|\mathcal{G}_j|} \leq \sqrt{|\mathcal{G}|}$$

und $b_{m,m} = |\mathcal{G}_{j-1}|/(|\mathcal{H}_1(j-1)||\mathcal{H}_2(j-1)|)$, woraus folgt

$$|\mathcal{H}_2(j)| = b_{m,m}|\mathcal{H}_2(j-1)| = \frac{b_{m,m}|\mathcal{G}_{j-1}|}{b_{m,m}|\mathcal{H}_1(j-1)|} = \frac{|\mathcal{G}_j|}{b_{j,j}|\mathcal{H}_1(j-1)|} \leq \sqrt{|\mathcal{G}_j|} \leq \sqrt{|\mathcal{G}|}.$$

Damit haben wir gezeigt, daß

$$|\mathcal{H}_i| \leq \sqrt{|\mathcal{H}|} = \sqrt{\prod_{i=1}^{j} b_{i,i}} \leq \sqrt{|\mathcal{G}|}, \quad i = 1, 2. \tag{8.13}$$

während der gesamten Laufzeit des Algorithmus gilt.

Sei nun $j \in \{1, \ldots, l\}$. Wir schätzen als nächstes ab, wie groß die restlichen Mengen, babySet, giantSet und auxiliaryBabySet, sind. Aus (8.9) und (8.10) folgt, daß

$$|\mathsf{auxiliaryBabySet}| = s|\mathcal{H}_1| = \left\lceil \sqrt{|\mathcal{H}|}/|\mathcal{H}_1| \right\rceil |\mathcal{H}_1| \leq 2\sqrt{|\mathcal{G}|} \tag{8.14}$$

und

$$|\mathsf{giantSet}| = t|\mathcal{H}_2| = \left\lceil \sqrt{|\mathcal{H}|}/|\mathcal{H}_2| \right\rceil |\mathcal{H}_2| \leq 2\sqrt{|\mathcal{G}|}. \tag{8.15}$$

Sei $e(j)$ die kleinste Zahl, für die (8.11) gilt. In Satz (8.1.2) haben wir bewiesen, daß

$$e(j) < \sqrt{2b_{j,j}} + 1/2 \leq 2\sqrt{b_{j,j}}.$$

Bei der Konstruktion der Menge babySet in der j-ten Iteration wird die Menge auxiliaryBabySet der $(j-1)$-ten Iteration verwendet. Deshalb gilt

$$|\mathsf{babySet}| \leq e(j)|\mathsf{auxiliaryBabySet}| \leq 4\sqrt{b_{j,j}}\sqrt{\prod_{i=1}^{j-1} b_{i,i}} \leq 4\sqrt{|\mathcal{G}|}.$$

Als nächstes bestimmen wir, wie oft der Algorithmus in einer Menge nach einem Tupel sucht. Da alle lesende Zugriffe auf die Mengen $\mathcal{H}_1$, $\mathcal{H}_2$, giantSet und auxiliaryBabySet sequenziell erfolgen, d.h. es werden alle Elemente nacheinander ausgelesen, sind nur die Zugriffe auf die Menge babySet von Interesse, denn bei diesen wird wirklich nach einem Tupel gesucht.

Wir unterscheiden zwei Fälle: $e(j) = 1$ und $e(j) > 1$. Im ersten Fall ist $b_{j,j} = 1$. Im zweiten Fall gilt $b_{j,j} \geq 2$. Aus Lemma 8.2.3 folgt, daß der Algorithmus HNFRelationBasis höchstens

$$\begin{aligned}
&\sum_{j=1}^{l} e(j)|\mathsf{giantSet}| \leq \sum_{j=1}^{l} 2e(j)\sqrt{\prod_{i=1}^{j-1} b_{i,i}} \leq \sum_{j=1}^{l} 4\prod_{i=1}^{j}\sqrt{b_{i,i}} \\
&\leq 4\sum_{j=1,e(j)>1}^{l}\prod_{i=1}^{j}\sqrt{b_{i,i}} + 4\sum_{j=1,e(j)=1}^{l}\prod_{i=1}^{j}\sqrt{b_{i,i}} \\
&\leq 4(2+\sqrt{2})\prod_{j=1,e(j)>1}^{l}\sqrt{b_{i,i}} + 4(l-l(M))\prod_{j=1}^{l}\sqrt{b_{i,i}} \\
&\leq 4(2+\sqrt{2}+l-l(M))\sqrt{|G|}
\end{aligned}$$

mal in der Menge babySet nach einem Tupel sucht.

Als nächstes schätzen wir die Anzahl der Gruppenmultiplikationen, die der Algorithmus ausführt ab. Diese Anzahl setzt sich zusammen aus den Multiplikationen

- zum Berechnen von $\mathsf{g} \cdot \mathsf{giantElement}$, mit $(\mathsf{g}, \mathbf{v}) \in \mathsf{giantSet}$

$$a_1 \leq \sum_{j=1}^{l} e(j)|\mathsf{giantSet}| \leq 4(2+\sqrt{2}+l-l(M))\sqrt{|\mathcal{G}|},$$

- zum Aktualisieren von babySet

$$a_2 \leq \sum_{j=1}^{l} e(j)|\mathsf{auxiliaryBabySet}| \leq 4(2+\sqrt{2}+l-l(M))\sqrt{|\mathcal{G}|},$$

- zum Berechnen von babyElement und giantElement

$$
\begin{aligned}
a_3 &\leq \sum_{j=1}^{l}(2e(j)-2) \leq 2\sum_{j=1}^{l} e(j) - 2l = 2\sum_{j=1,e(j)>1}^{l} e(j) + 2\sum_{j=1,e(j)=1}^{l} e(j) - 2l \\
&\leq 2\sum_{j=1,e(j)>1}^{l} (\sqrt{2b_{j,j}} + 1/2) + 2(l - l(M)) - 2l \\
&\leq 2\sqrt{2}\sum_{j=1,e(j)>1}^{l)} \sqrt{b_{j,j}} + l(M) - 2l(M) \\
&\leq 2\sqrt{2}\prod_{j=1,e(j)>1}^{l} \sqrt{b_{j,j}} + 2\sqrt{2}(l(M)-1) - l(M) \quad \text{(Lemma 8.2.3)} \\
&\leq 2\sqrt{2|\mathcal{G}|} + 2l(M)
\end{aligned}
$$

und

- zum Aktualisieren von $\mathcal{H}_1$, $\mathcal{H}_2$, auxiliaryBabySet und giantSet. In diesem Fall werden keine Multiplikationen ausgeführt, wenn $b_{j,j} = 1$ ist. In jeder **loop**-Schleife mit $b_{j,j} > 1$ werden entweder $|\mathcal{H}_1|$ Multiplikationen zum Aktualisieren von $\mathcal{H}_1$ oder $|\mathcal{H}_2|$ Multiplikationen zum Aktualisieren von $\mathcal{H}_2$ ausgeführt. Außerdem werden |auxiliaryBabySet| + |giantSet| Multiplikationen zum Aktualisieren von auxiliaryBabySet und giantSet und höchstens $2\left\lfloor \log\sqrt{|\mathcal{G}|}\right\rfloor$ Multiplikationen zum Berechnen von g_m^s während der Aktualisierung von giantSet ausgeführt. Zusammen sind das höchstens

$$
a_4 \leq \sum_{j=1}^{l}(5\sqrt{\prod_{i=1}^{j} b_{i,i}} + 2\lfloor\log\sqrt{|G|}\rfloor) \leq 5(2+\sqrt{2})\sqrt{|G|} + 2l(M)\log\sqrt{|G|}.
$$

Gruppenmultiplikationen.

Die Anzahl der Gruppenmultiplikationen, die der Algorithmus HNFRelationBasis ausführt ist also höchstens

$$
a_1 + a_2 + a_3 + a_4 \leq (48 + 8l - 6l(M))\sqrt{|\mathcal{G}|} + 2l(M)\log\sqrt{|\mathcal{G}|}
$$

Zum Schluss bestimmen wir noch, wie oft HNFRelationBasis Elemente aus $\mathcal{G}$ invertiert. Dies geschieht genau l mal, um das Inverse von g_j, $1 \leq j \leq l$, aus M zu berechnet.

Die Gruppenmultiplikationen und -invertierungen ergeben zusammen die Anzahl der Gruppenoperationen aus dem Satz. □

8.2.2 Berechnung der Invarianten

Nun präsentieren wir den vollständigen Algorithmus zur Berechnung der Struktur einer endlichen abelschen Gruppe.

Algorithm 51 STRUCTURE(M)

Input: Ein Erzeugendensystem $M = (\mathsf{g}_1, \ldots, \mathsf{g}_l)$ für $\mathcal{G}$
Output: Invarianten $(n_1, \ldots, n_k)$ von $\mathcal{G}$ und $\mathsf{h}_1, \ldots, \mathsf{h}_k \in \mathcal{G}$ mit $|\langle \mathsf{h}_i \rangle| = n_i$ für $1 \leq i \leq k$ und $\mathcal{G} \cong \langle \mathsf{h}_1 \rangle \times \cdots \times \langle \mathsf{h}_k \rangle$

```
B ← HNFRelationBasis(M)
((d_{i,j}), (u_{i,j}), (v_{i,j})) ← SNF(B)
for j = 1, ..., k do
    h_j ← ∏_{i=1}^{l} g_i^{u_{i,j}}
return((d_{1,1}, ..., d_{k,k}), (h_1, ..., h_k))
```

Satz 8.2.5 *Bei Eingabe eines Erzeugendensystems M einer endlichen abelschen Gruppe $\mathcal{G}$ berechnet* STRUCTURE *die Invarianten $n_1, \ldots, n_k$ von $\mathcal{G}$ und Elemente $\mathsf{h}_1, \ldots, \mathsf{h}_k \in \mathcal{G}$, so daß $\mathcal{G} \cong \langle \mathsf{h}_1 \rangle \times \cdots \times \langle \mathsf{h}_k \rangle$ und $|\langle \mathsf{h}_i \rangle| = n_i$ für $1 \leq i \leq k$. Der Algorithmus führt $O(|M|\sqrt{|\mathcal{G}|})$ Gruppenoperationen und Vergleiche durch. Zusätzlich werden $O(|M|^3 \log|\mathcal{G}| \log\log|\mathcal{G}| \log\log\log|\mathcal{G}|)$ Bitoperationen zum Berechnen der Smith-Normalform ausgeführt. Die Anzahl der Zwischengepeicherten Gruppenelemente ist $O(\sqrt{|\mathcal{G}|})$.*

Beweis. Wenn die Tabellen, in denen Gruppenelemente zwischengepeichert und gesucht werden, als Hash-Tabellen implementiert werden, so daß die Zeit zum Suchen $O(1)$ ist, dann folgt der Beweis des Satzes sofort aus Lemma 8.2.1, Satz 8.2.4 und Satz 2.4.17.

Kapitel 9

Ausblick

In der vorliegenden Arbeit wurden Algorithmen für Quantencomputer zur Lösung zahlentheoretischer Probleme vorgestellt und untersucht. Es ist ein erster Versuch, diese Probleme mit Hilfe von Quantencomputer zu lösen und so scheinen diese Lösungen noch nicht optimal zu sein. In diesem letzten Kapitel soll eine Übersicht über potenziellen Verbesserungen sowie weitere offene Probleme gegeben werden.

Im Kapitel 4 wurden Algorithmen für imaginär-quadratische Zahlkörper vorgestellt. Dabei wurde beim Reduktionsalgorithmus ein einfacher Ansatz verwendet, indem das Reduktionsoperator ρ $O(|\Delta|)$ mal aufgerufen wurde. Die Gesamtkomplexität des Algorithmus ist deshalb $O(|\Delta|^3)$ bzw. $O(|\Delta|^4)$, falls Quantenaddition verwendet wird. Aus dem klassischen Computermodell ist jedoch bekannt, daß die Reduktion in Zeit $O(|\Delta|^2)$ möglich ist (siehe [BV07]), wenn man die Formeln aus dem Kapitel 2 verwendet. Um diese Laufzeit mit einem Quantencomputer zu erreichen, kann die Idee aus [PZ03] (siehe auch XGCD-Algorithmus) zur Desynchronisierung der Berechnung verwendet werden.

Im Kapitel 5 wurden Algorithmen für reell-quadratische Zahlkörper vorgestellt. Der Schwerpunkt lag dabei auf der Untersuchung der benötigten Qubits. Interessant wäre hier festzustellen, wie viele elementare Quantengatter für die Ausführung von präsentierten Algorithmen benötigt werden.

Ein anderes Problem aus der Zahlentheorie ist die Bestimmung der Klassengruppe. Mit Ansatz von Kitaev (siehe [Kit96]) läßt sich die Klassengruppe imaginär-quadratischer Zahlkörper einfach berechnen. Schwieriger ist der Fall in reell-quadratischen Körpern und in Körpern mit Grad größer zwei. Wegen der Approximationen reeller Zahlen läßt sich Kitaevs Ansatz nicht eins zu eins übertragen. Hier muß noch untersucht werden, ob und wie Kitaevs Ansatz oder Shors Ansatz, wie im Quantenframework aus Kapitel 3 beschrieben, diese Probleme lösen können.

Eine weite Herausforderung ist zu untersuchen, auf welche weitere Probleme sich das Quantenframework aus Kapitel 3 anwenden läßt. Interessant wäre zu untersuchen, ob damit Kryptoverfahren gebrochen werden können, die auf Fehler-korrigierenden Codes (z.B. McEliece) basieren. Hierzu kann eventuell die Periodizität der zugrunde liegenden Goppa-Polynome ausgenutzt werden.

Beim Laufzeitvergleich der Kryptosysteme, haben wir uns auf einige wenige beschränkt. Ein Vergleich mit anderen Kryptosystemen (z.B. McEliece [McE78], [EOS07] oder NTRU [HPS98]), welche auf anderen mathematischen Primitiven beruhen, muß noch Durchgeführt werden. Für diese alternativen Kryptosysteme sind bis jetzt noch keine Quantenalgorithmen bekannt, die sie wesentlich schneller als klassische Algorithmen brechen können. So kann dann der Zeitpunkt bestimmt werden, ab welcher Größe der Quantencomputer die alternativen Verfahren vorzuziehen sind.

Die wichtigste offene Frage ist jedoch die Frage: Wie baut man große Quantencomputer?

Literaturverzeichnis

[Abe94] Christine Abel. *Ein Algorithmus zur Berechnung der Klassenzahl und des Regulators reellquadratischer Ordnungen*. PhD thesis, Universität des Saarlandes, Saarbrücken, Germany, 1994. German.

[Art93] Michael Artin. *Algebra*. Birkhäuser Verlag, Basel, Boston, Berlin, 1993.

[Ban93] W. Banaszczyk. New bounds in some transference theorems in the geometry of numbers. *Mathematische Annalen*, 296:625–635, 1993.

[BB94] I. Biehl and J. Buchmann. Algorithms for quadratic orders. In *Proceedings of Symposia in Applied Math.*, volume 48, pages 425–449, 1994.

[Bea02] Stephane Beauregard. Circuit for shor's algorithm using 2n+3 qubits. `arXiv: quant-ph/0205095`, 2002.

[Ben77] C.H. Bennett. Logical reversibility of computation. *IBM Journal of Research and Development*, pages 525–532, November 1977.

[BJT97] Johannes Buchmann, Michael J. Jacobson, Jr., and Edlyn Teske. On some computational problems in finite abelian groups. *Mathematics of Computation*, 66(220):1663–1687, 1997.

[BK93] J. Buchmann and V. Kessler. Computing a reduced lattice basis from a generating system. Technical Report ??/93, Technische Universität Darmstadt, Fachbereich Informatik, 1993. `http://www.informatik.tu-darmstadt.de/TI/Veroeffentlichung/TR/`.

[BP89] J. Buchmann and M. Pohst. Computing a lattice basis from a system of generating vectors. In James H. Davenport, editor, *EUROCAL 1987*, volume 378 of *Lecture Notes in Computer Science*, pages 54–63. Springer, 1989.

[BS05] Johannes Buchmann and Arthur Schmidt. Computing the structure of a finite abelian group. *Mathematics of Computation*, 74(252):2017–2026, 2005.

[Buc87] J. Buchmann. On the period length of the generalized Lagrange algorithm. *Journal of Number Theory*, 26:31–37, 1987.

[Buc88] J. Buchmann. Zur Komplexität der Berechnung von Einheiten und Klassenzahlen algebraischer Zahlkörper. Habilitationsschrift, Universität Düsseldorf, Germany, 1988.

[BV07] Johannes Buchmann and Ulrich Vollmer. *Algorithms for binary quadratic forms.* Springer Verlag, 2007.

[Coh00] H. Cohen. *A course in computational algebraic number theory.* Springer, Heidelberg, 2000.

[CVZ+98] I. Chuang, L. Vandersypen, X. Zhou, D. Leung, and S. Lloyd. Experimental realization of a quantum algorithm. *Nature*, 393:143–146, 1998.

[Dra00] Thomas G. Draper. Addition on a quantum computer. `http://arxiv.org/abs/quant-ph/0008033`, August 07 2000.

[EOS07] D. Engelbert, R. Overbeck, and A. Schmidt. A summary of McEliece-type cryptosystems and their security. *Journal of Mathematical Cryptology*, 1(2), 2007.

[GMP] GNU multiple precision arithmetic library 4.1.4. `http://swox.com/gmp/`.

[GN96] Robert B. Griffiths and Chi-Sheng Niu. Semiclassical fourier transform for quantum computation. *Phys. Rev. Lett.*, 76(17):3228–3231, Apr 1996.

[Hal02] Sean Hallgren. Polynomial-time quantum algorithms for pell's equation and the principal ideal problem. In *Proceedings of the thiry-fourth annual ACM symposium on the theory of computing*, pages 653–658. ACM Press, 2002.

[Ham] Safuat Hamdy. `libiq` — a library for arithmetic in class groups of imaginary quadratic orders. `http://www.math.ucalgary.ca/~hamdy/libiq.html`.

[Ham02] Safuat Hamdy. *Über die Sicherheit und Effizienz kryptografischer Verfahren mit Klassengruppen imaginär-quadratischer Zahlkörper.* PhD thesis, Technische Universität Darmstadt, Fachbereich Informatik, Darmstadt, Germany, 2002. `http://www.informatik.tu-darmstadt.de/ftp/pub/TI/reports/hamdy.diss.pdf`.

[HHR+05] H. Häffner, W. Hänsel, C. F. Roos, J. Benhelm, D. Chek al kar, M. Chwalla, T. Körber, U. D. Rapol, M. Riebe, P. O. Schmidt, C. Becher, O. Gühne, W. Dür, and R. Blatt. Scalable multiparticle entanglement of trapped ions p643. *Nature*, 438:643 – 646, December 2005.

[HM91] J.L. Hafner and K.S. McCurley. Asymptotically fast triangularization of matrices over rings. *SIAM J. Comput.*, 20:1068–1083, 1991.

[Høy99] Peter Høyer. Conjugated operators in quantum algorithms. *Phys. Rev. A*, 59(5):3280–3289, May 1999.

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In Joe P. Buhler, editor, *ANTS*, volume 1423 of *Lecture Notes in Computer Science*, pages 267–288. Springer-Verlag, 1998.

[Hua82] Loo-Keng Hua. *Introduction to Number Theory.* Springer, New York, 1982.

[JSW06] Michael J. Jacobson Jr., Renate Scheidler, and Hugh C. Williams. An improved real-quadratic-field-based key exchange procedure. *J. Cryptology*, 19(2):211–239, 2006.

[Kit96] Alexei Kitaev. Quantum measurements and the abelian stabilizer problem. *Electronic Colloquium on Computational Complexity (ECCC)*, 3(3), 1996.

[Knu81] D.E. Knuth. *The art of computer programming. Volume 2: Seminumerical algorithms*. Addison-Wesley, Reading, Massachusetts, 1981.

[Lan02] Serge Lang. *Algebra*. Springer, New York, 2002.

[Len82] Hendrik W. Lenstra, Jr. On the calculation of regulators and class numbers of quadratic fields. In J. V. Armitage, editor, *Journees Arithmetiques, Exeter 1980*, volume 56 of *London Mathematical Society Lecture Notes Series*, pages 123–150. Cambridge University Press, 1982.

[Len92] H.W. Lenstra Jr. Algorithms in algebraic number theory. *Bull. Amer. Math. Soc. (N.S.)*, 26:211–244, 1992.

[LiD] *LiDIA — A C++ Library For Computational Number Theory*. The LiDIA Group.

[LLL82] A.K. Lenstra, H.W. Lenstra Jr., and C. Lovasz. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:515–534, 1982.

[Lud05] Christoph Ludwig. *Practical Lattice Basis Sampling Reduction*. PhD thesis, Technische Universität Darmstadt, Fachbereich Informatik, Darmstadt, 2005.

[LV01] Arjen K. Lenstra and Eric R. Verheul. Selecting cryptographic key sizes. *Journal of Cryptology: the journal of the International Association for Cryptologic Research*, 14(4):255–293, 2001.

[Mau00] Markus Maurer. *Regulator approximation and fundamental unit computation for real quadratic orders*. PhD thesis, Technische Universität Darmstadt, Fachbereich Informatik, Darmstadt, Germany, 2000.

[McE78] R.J. McEliece. A public key cryptosystem based on algebraic coding theory. *DSN progress report*, 42-44:114–116, 1978.

[ME99] Michele Mosca and Artur Ekert. The hidden subgroup problem and eigenvalue estimation on a quantum computer. *Lecture Notes in Computer Science*, 1509:174–188, 1999.

[MG02] D. Micciancio and S. Goldwasser. *Complexity of Lattice Problems*. Kluwer Academic Publishers, Norwell, MA, USA, 2002.

[NC00] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[PB99] Igor Pak and Sergey Bratus. On sampling generating sets of finite groups and product replacement algorithm: extended abstract. In *ISSAC '99: Proceedings of the 1999 international symposium on Symbolic and algebraic computation*, pages 91–96, New York, NY, USA, 1999. ACM Press.

[Per57] Oskar Perron. *Die Lehre von den Kettenbrüchen*. Teubner, Stuttgart, 3. verb. u. erw. Auflage, 1954-1957.

[PZ03] J. Proos and C. Zalka. Shor's discrete logarithm quantum algorithm for elliptic curves. `http://arxiv.org/abs/quant-ph/0301141`, 2003.

[Saw04] Reg Sawilla. Fast ideal arithmetic in quadratic fields. Master's thesis, University of Calgary, 2004.

[Sch] C. P. Schnorr. Gittertheorie und algorithmische Geometrie.

[Sch06] Arthur Schmidt. Quantum algorithm for solving the discrete logarithm problem in the class group of an imaginary quadratic field and security comparison of current cryptosystems at the beginning of quantum computer age. In Günter Müller, editor, *ETRICS*, volume 3995 of *Lecture Notes in Computer Science*, pages 481–493. Springer, 2006.

[Sho94] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *IEEE Symposium on Foundations of Computer Science*, pages 124–134, 1994.

[SJ] Reg Sawilla and Michael J. Jacobson. private kommunikation.

[Sla69] I. S. Slavutskii. Upper bounds and numerical calculation of the number of ideal classes of real quadratic fields. *Amer. Math. Soc. Transl.*, 82(2):67–71, 1969.

[SV05] Arthur Schmidt and Ulrich Vollmer. Polynomial time quantum algorithm for the computation of the unit group of a number field. In Harold N. Gabow and Ronald Fagin, editors, *STOC*, pages 475–480. ACM, 2005.

[Ter00] David C. Terr. A modification of Shanks' baby-step giant-step algorithm. *Mathematics of Computation*, 69(230):767–773, 2000.

[Thi95] Christoph Thiel. *On the complexity of some problems in algorithmic algebraic number theory.* PhD thesis, Universität des Saarlandes, Saarbrücken, Germany, 1995.

[VAE96] V. Vedral, A.Barenco, and A. Ekert. Quantum networks for elementary arithmetic operations. *Physical Review A*, 54:147–153, 1996.

[Vol03] Ulrich Vollmer. *Invariant and Discrete Logarithm Computation in Quadratic Orders.* PhD thesis, Technische Universität Darmstadt, Fachbereich Informatik, 2003.

[VSB+00] L. M. K. Vandersypen, M. Steffen, G. Breyta, C. S. Yannoni, R. Cleve, and I. L. Chuang. Experimental realization of an order-finding algorithm with an NMR quantum computer. *Physical Review Letters*, 85:5452–5455, 2000.

[VSB+01] L. M. K. Vandersypen, M. Steffen, G. Breyta, C. S. Yannoni, M. H. Sherwood, and I. L. Chuang. Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance. *Nature*, 414:883–887, 2001.